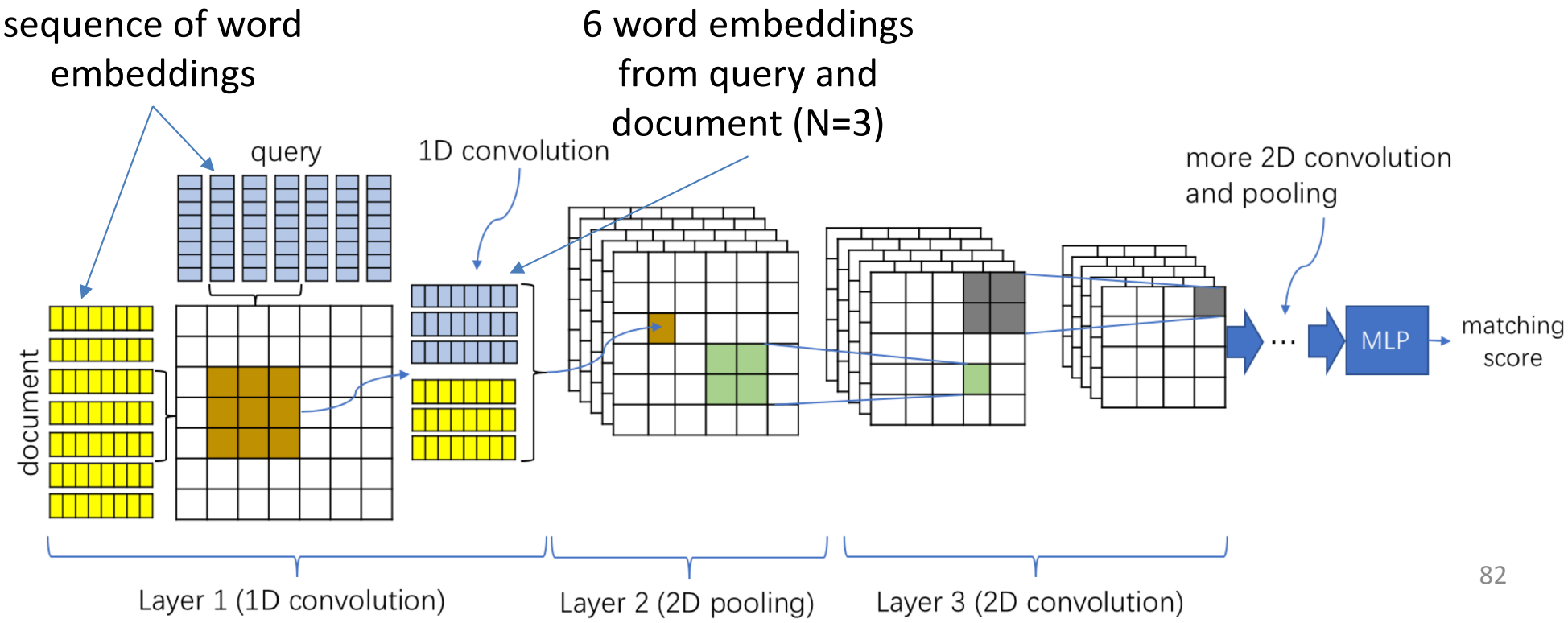
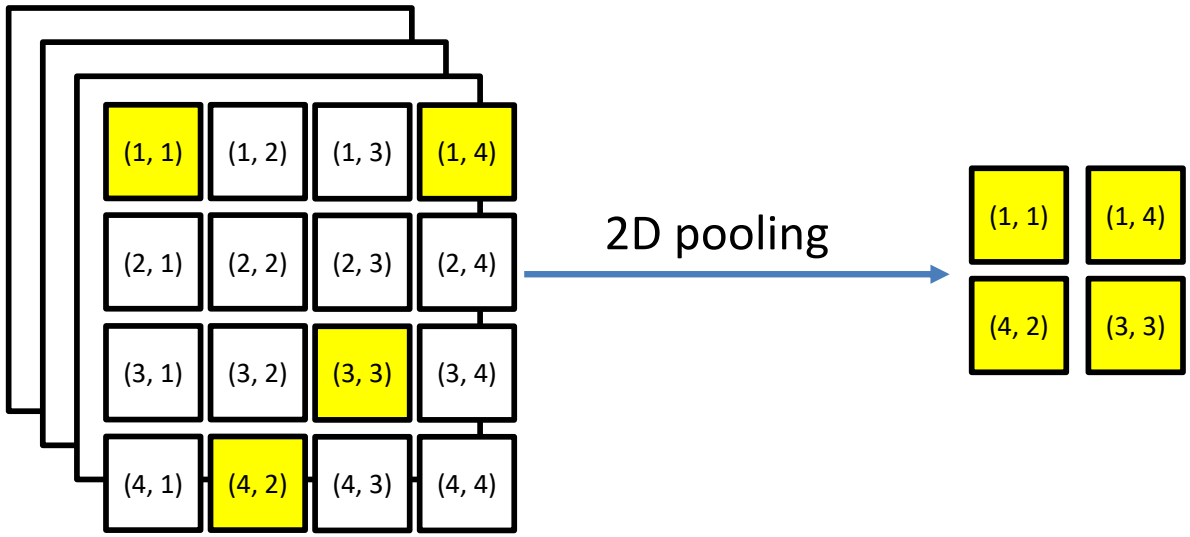
ARC-II (Hu et al., NIPS '14)

- Let two sentences meet before their own high-level representations mature
- Basic matching signals: phrase sum interaction matrix
- Interaction: CNN to capture the local interaction structure
- Aggregation function: MLP



ARC-II (cont')

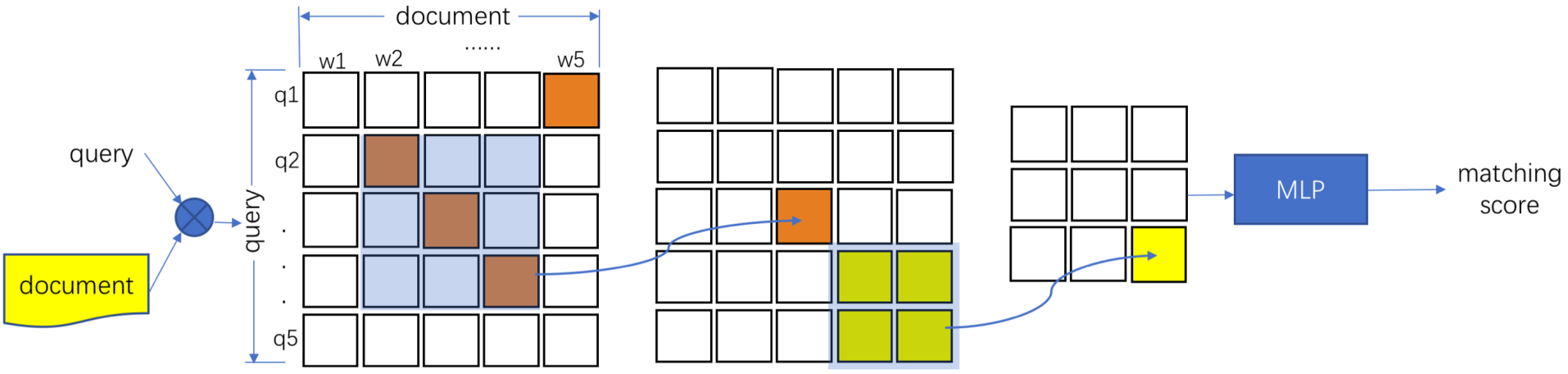
- Keeping word order information
  - Both the convolution and pooling are order preserving



- However, word level exact matching signals are lost
  - 2-D matching matrix is constructed based on the embedding of the words in two N-grams

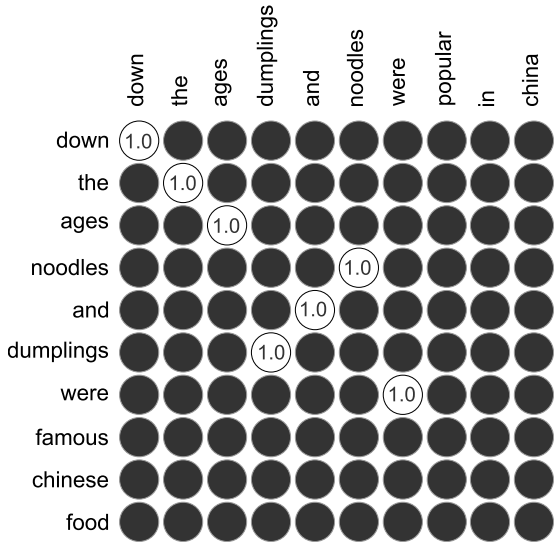
MatchPyramid (Pang et al., AAAI '16)

- Inspired by image recognition
- Basic matching signals: word-level matching matrix
- Matching function: 2D convolution + MDP

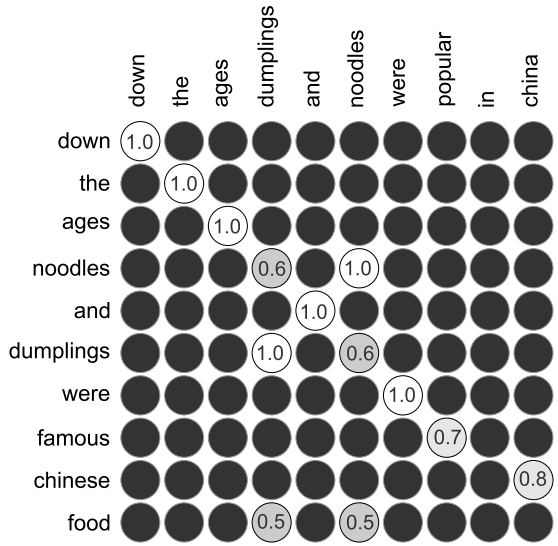


Matching Matrix: Basic Matching Signals

- Each entry calculated based on
  - Word-level exact matching (0 or 1)
  - Semantic similarity based on embeddings of words



Exact match

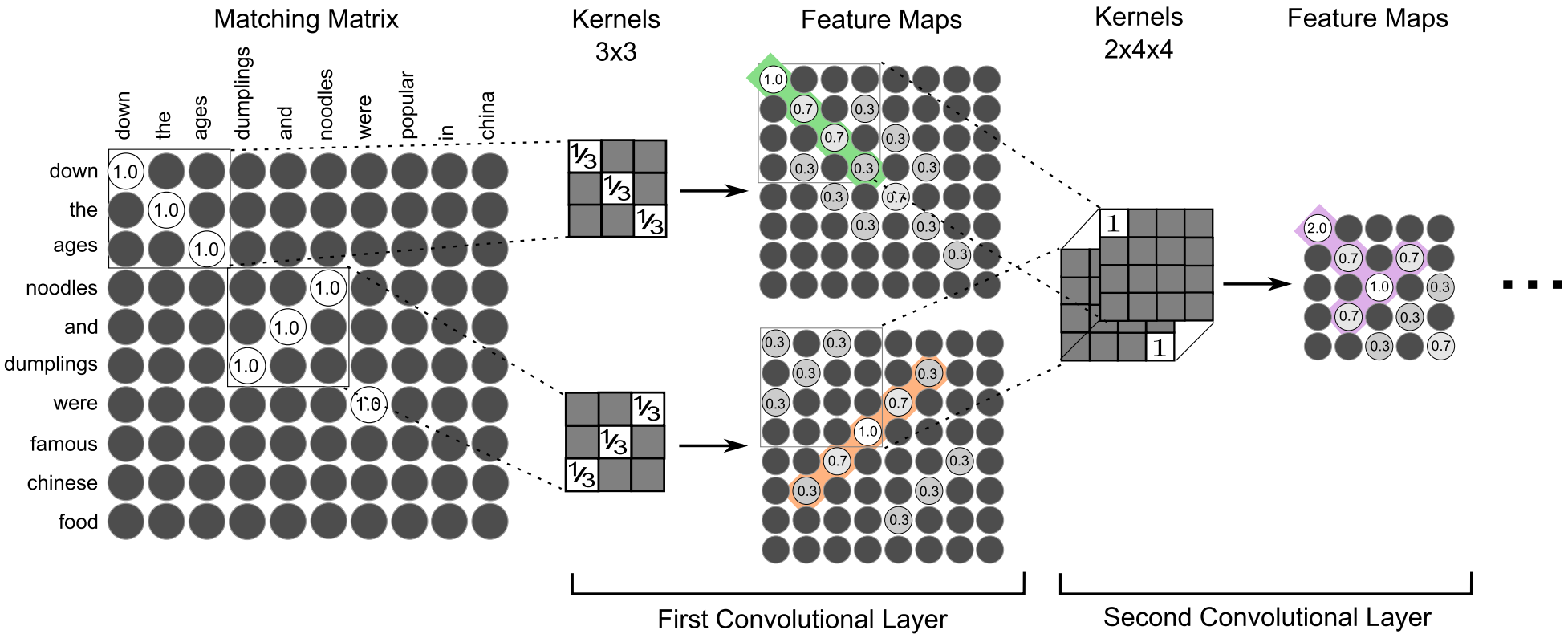


Cosine similarity

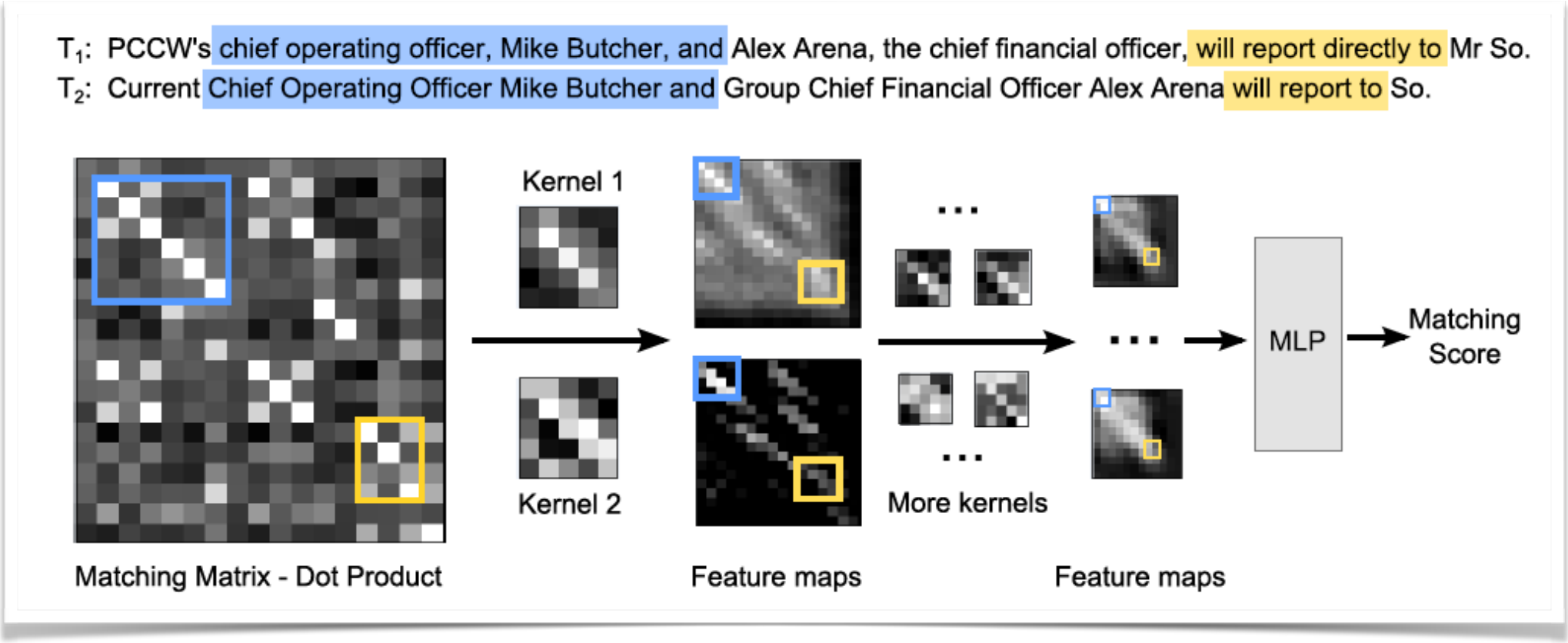
- Positions information of words is kept

Matching Function: 2D Convolution

- Discovering the matching patterns with CNN, stored in the kernels

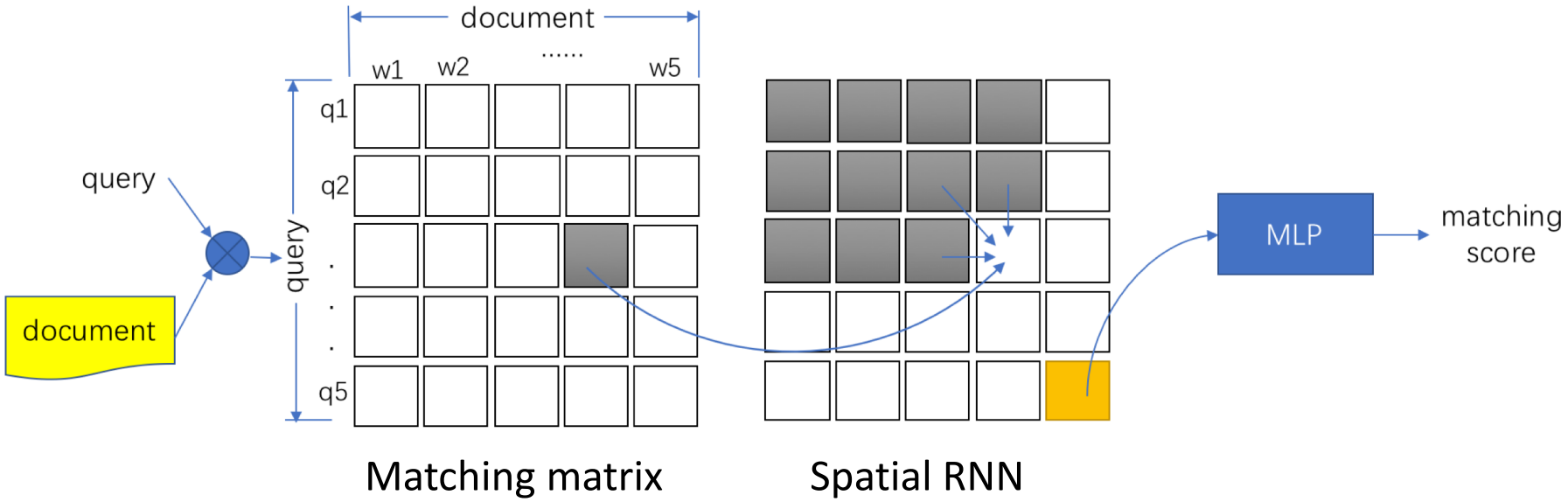


Discovered Matching Patterns

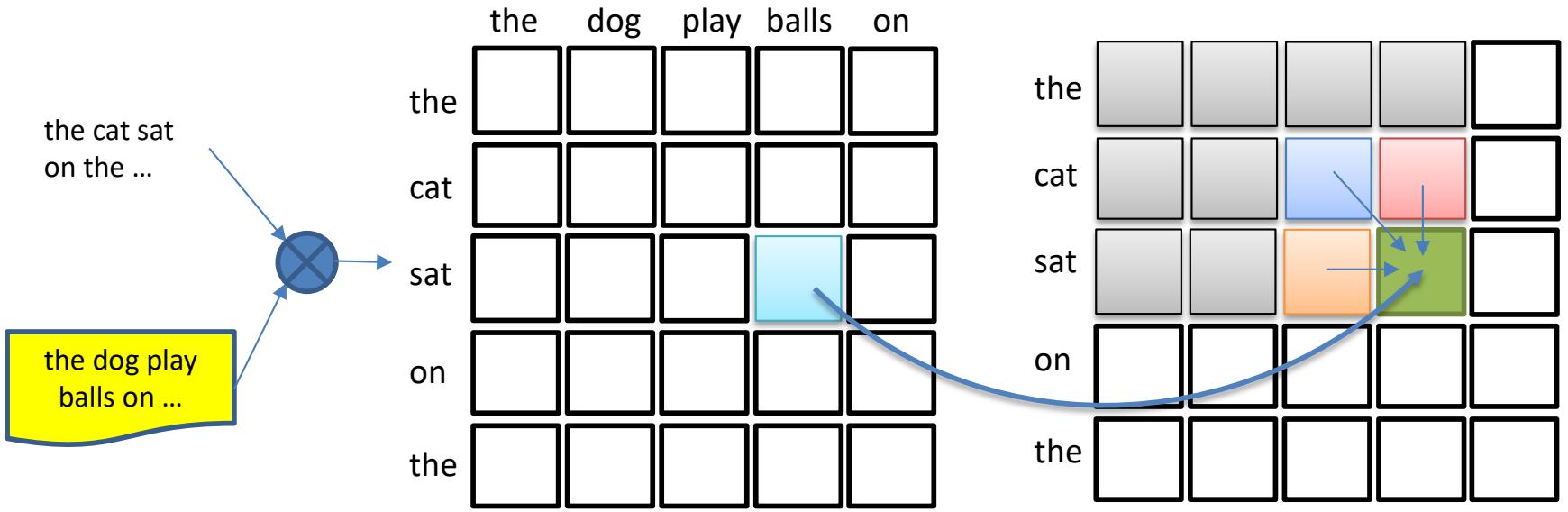


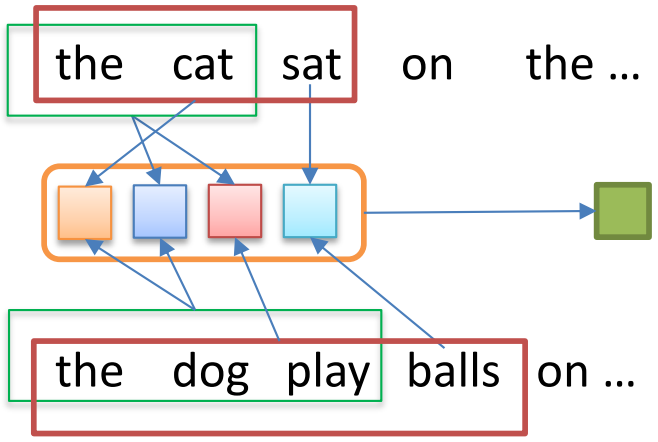
Match-SRNN (Wan et al., IJCAI '16)

- Based on spatial recurrent neural network (SRNN)
- Basic matching signals: word-level matching matrix
- Matching function: Spatial RNN + MLP



Match-SRNN: Recursive Matching Structure

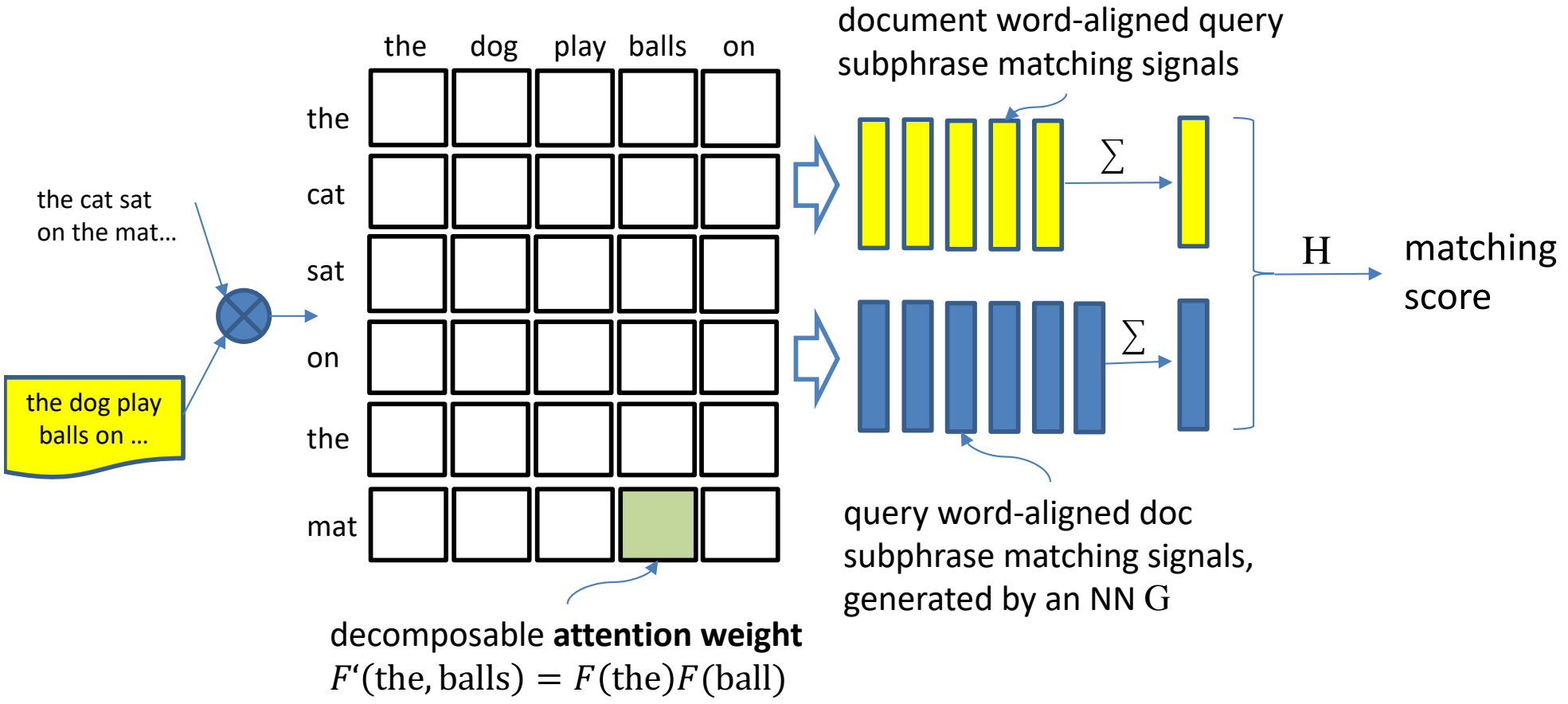




- Calculated recursively (from top left to bottom right)
- All matching signals between the prefixes been utilized
  - **Current position**: sat <—> balls
  - **Substrings**:
    - the cat <—> the dog play
    - the cat <—> the dog play balls
    - the cat sat <—> the dog play

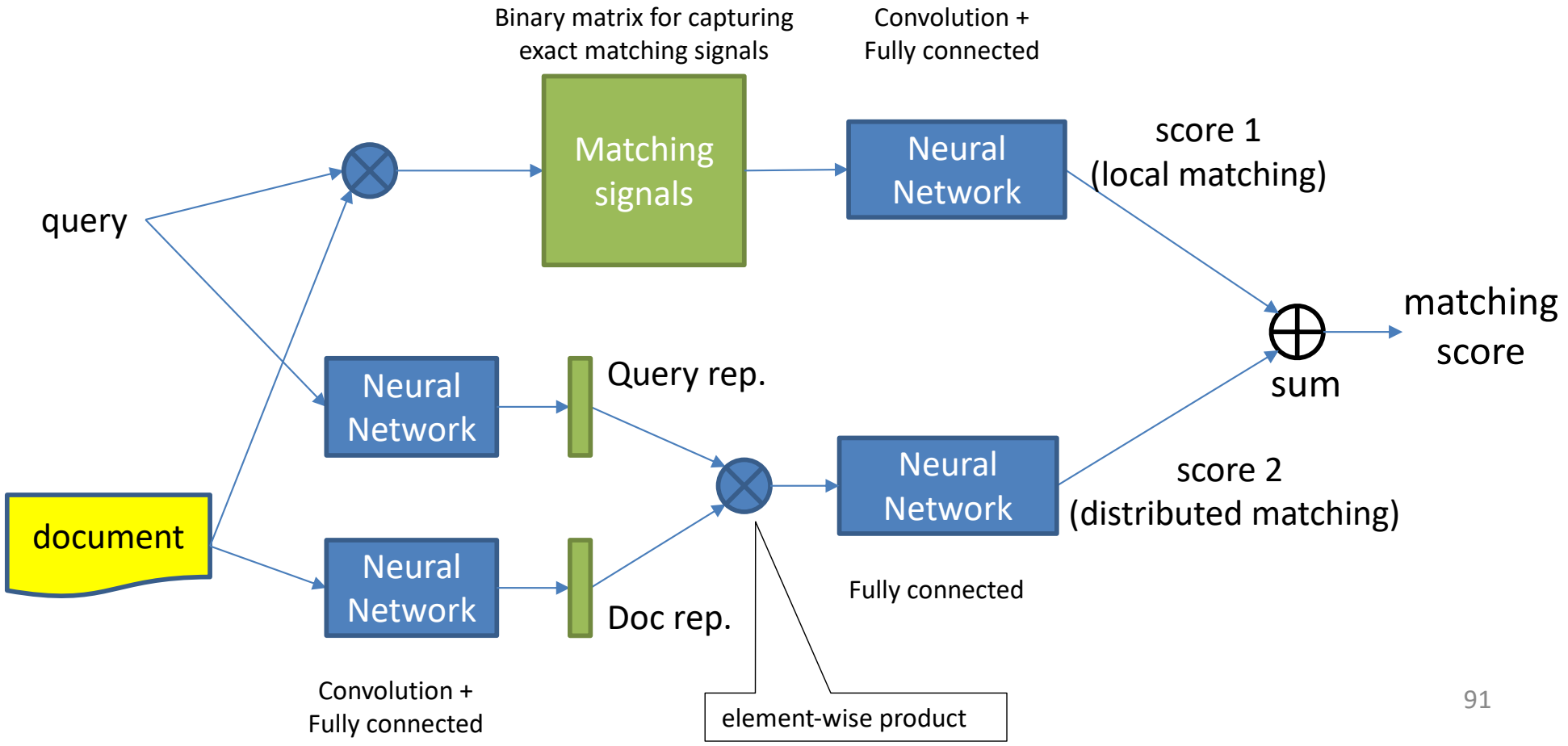
Decomposable Attention Model for Matching (Parikh et al., EMNLP '16)

- Based on decomposable attention model
- Three steps: attend -> compare -> aggregate
  - **Attend**: soft-align words of query and document
  - **Compare**: separately compare word-aligned subphrase, get matching signals
  - **Aggregate**: aggregate the matching signals for produce final matching score



Representation Learning + Matching Function Learning (Duet, Mitra et al., WWW '17)

- Hypothesis: matching with distributed representations complements matching with local representations
  - Local matching: matching function learning
  - Distributed matching: representation learning



Experimental Evaluation

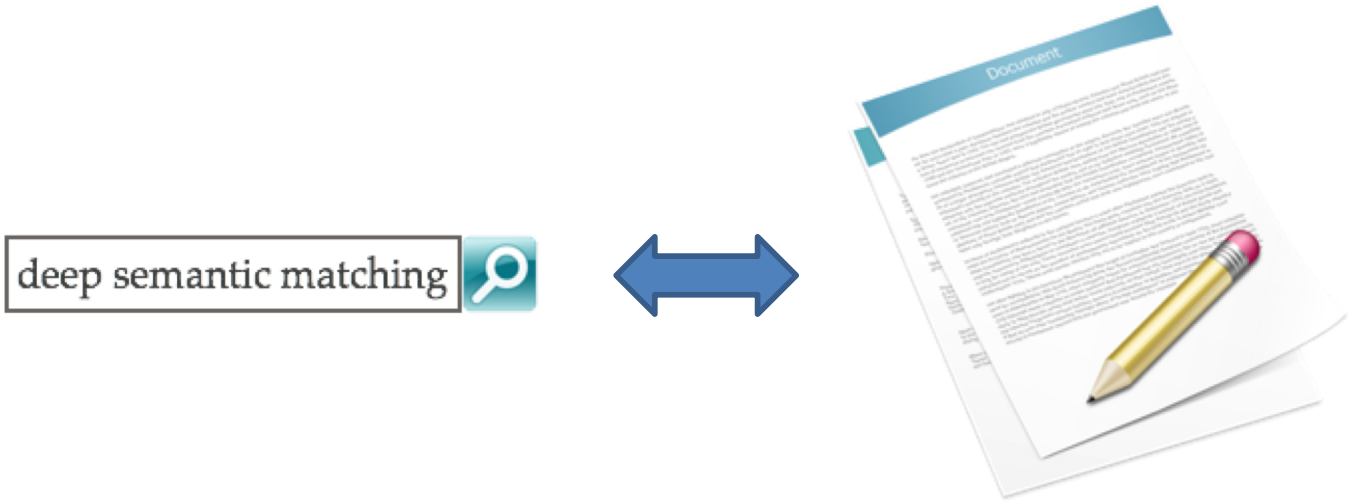
	Method	P@1	MRR
Traditional IR	BM25	0.579	0.457
Representation Learning methods	ARC-I	0.581	0.756
	CNTN	0.626	0.781
	LSTM-RNN	0.690	0.822
Matching Function Learning	ARC-II	0.591	0.765
	MatchPyramid	0.764	0.867
	Match-SRNN	0.790	0.882

Based on Yahoo! Answers dataset (60,564 question-answer pairs)

- Matching function learning based methods outperformed the representation learning ones

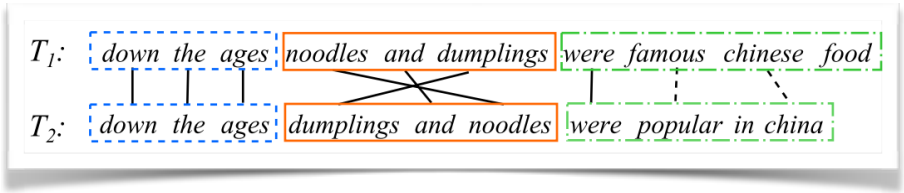
Short Summary

- Two steps
  - 1. Construct basic matching signals
  - 2. Aggregate matching patterns
- Basic matching signals
  - Similarity matrix (exact match, dot product, cosine similarity)
  - Attention weights
- Aggregate matching patterns
  - CNN
  - Spatial RNN
  - MLP
- Combining representation learning (inexact match) and matching function learning (exact match)



QUERY-DOCUMENT RELEVANCE MATCHING

Similarity ≠ Relevance (Pang et al., Neu-IR workshop '16)



Similarity matching

- Whether two sentences are semantically similar
- Homogeneous texts with comparable lengths
- Matches at all positions of both sentences
- Symmetric matching function
- Representative task: Paraphrase Identification

Relevance matching

- Whether a document is relevant to a query
- Heterogeneous texts (keywords query, document) and very different in lengths
- Matches in different parts of documents
- Asymmetric matching function
- Representative task: ad-hoc retrieval

Typical Query-Document Relevance Matching Methods

- Based on global distribution of matching strengths
  - DRMM (Guo et al., CIKM ’16)
  - aNMM (Yang et al., CIKM ‘16)
  - K-NRM (Xiong et al., SIGIR ‘17)
  - Conv-KNRM (Dai et al., WSDM ‘18)
- Based on local context of matched terms
  - DeepRank (Pang et al., CIKM ‘17)
  - PACRR (Hui et al., EMNLP ’17)

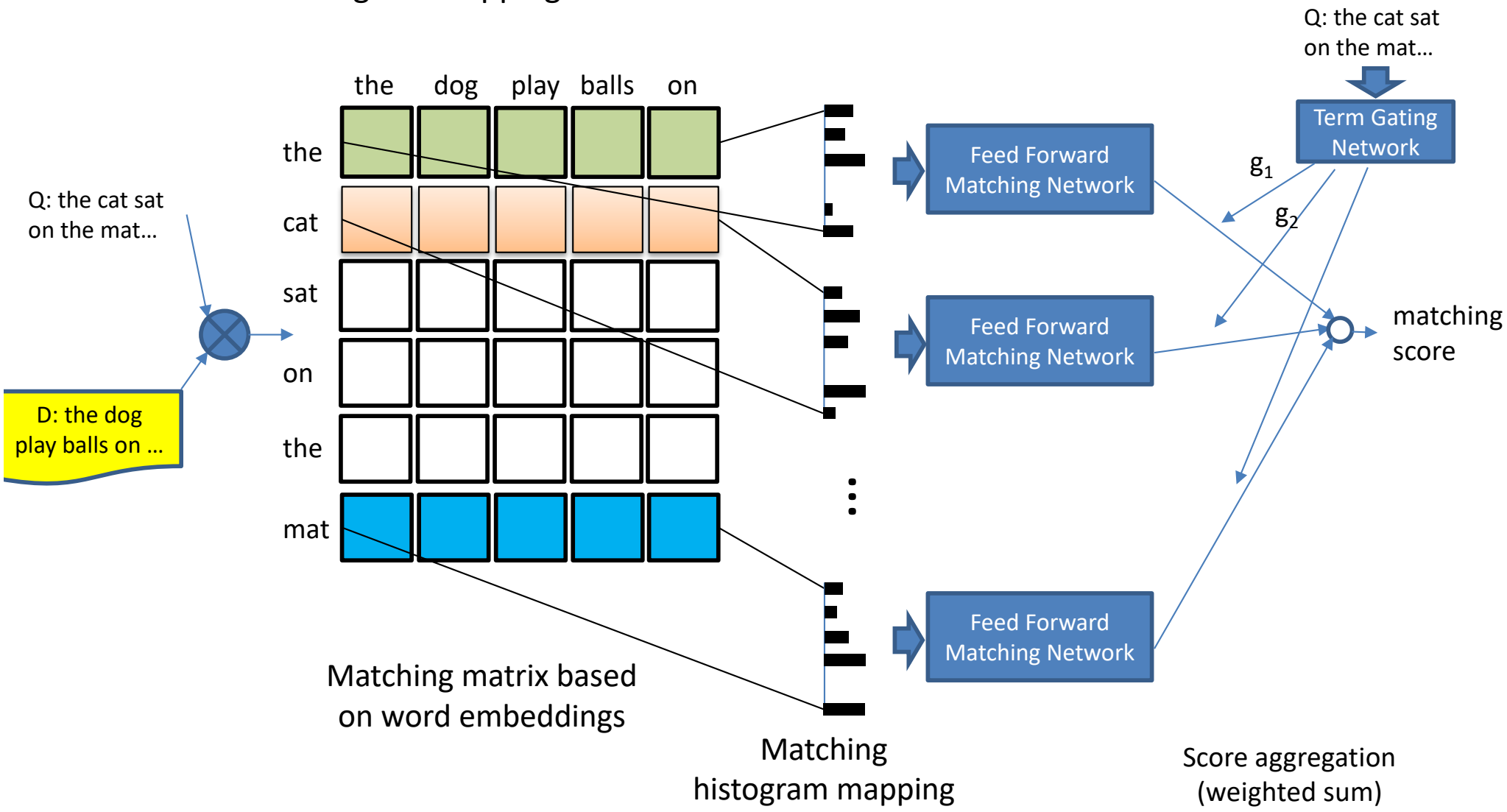
Relevance Matching based on Global Distribution of Matching Strengths

- Step 1: for each query term
  - Calculate its matching signals among the document
  - Calculate the global matching strength distributions
- Step 2: aggregate the distributions

- Advantages
  - Conducting matching between **short query** and **long document**
  - Matching strength distributions are robust, compared with the raw matching signals
- Disadvantage
  - **Lost term order** information when calculating the distributions

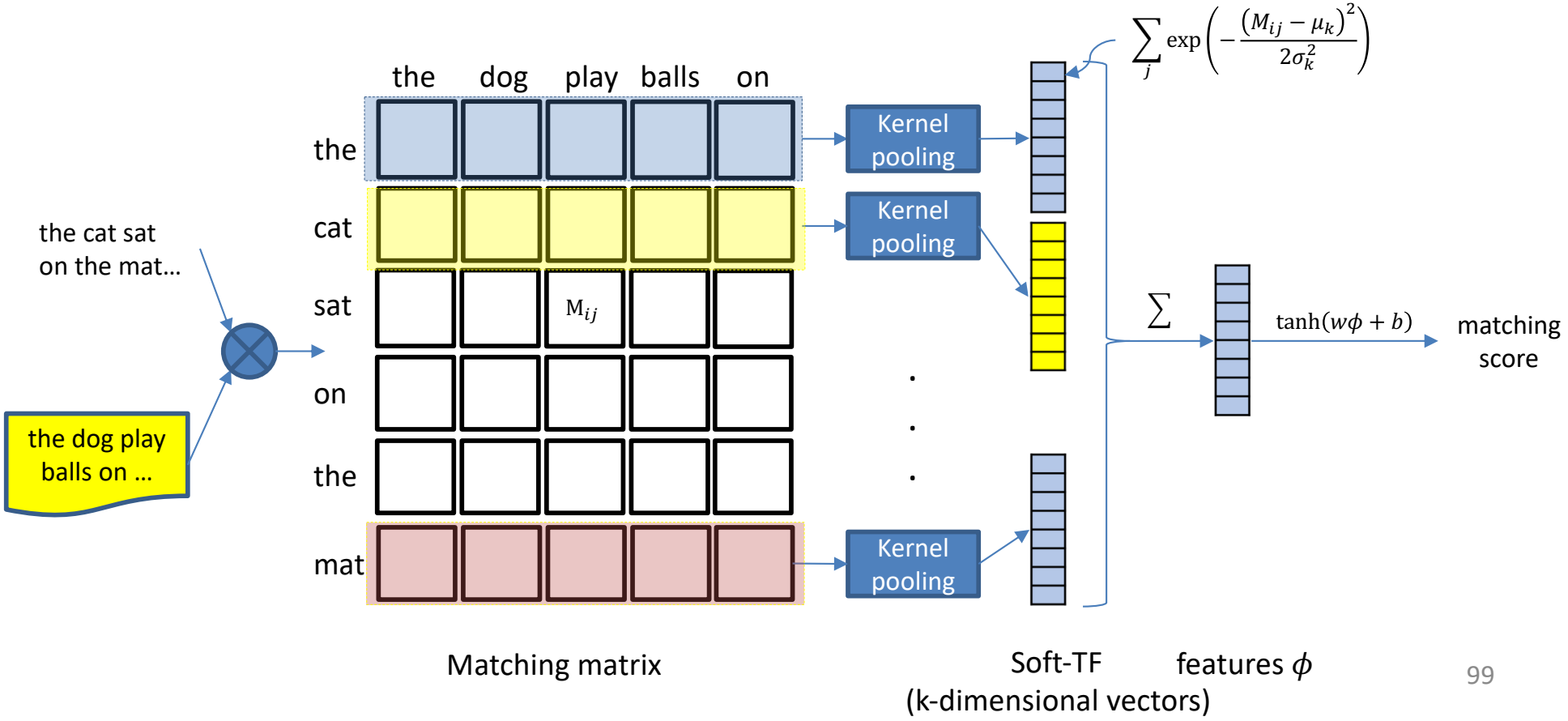
Deep Relevance Matching Model (Guo et al., CIKM '16)

- Basic matching signals: cosine similarity of word embeddings → matching strength histogram
- Ranking function: Neural Network + Term Gating Network
- Semantic gap: embeddings bridge the semantic gap
- Word order: histogram mapping lost order information



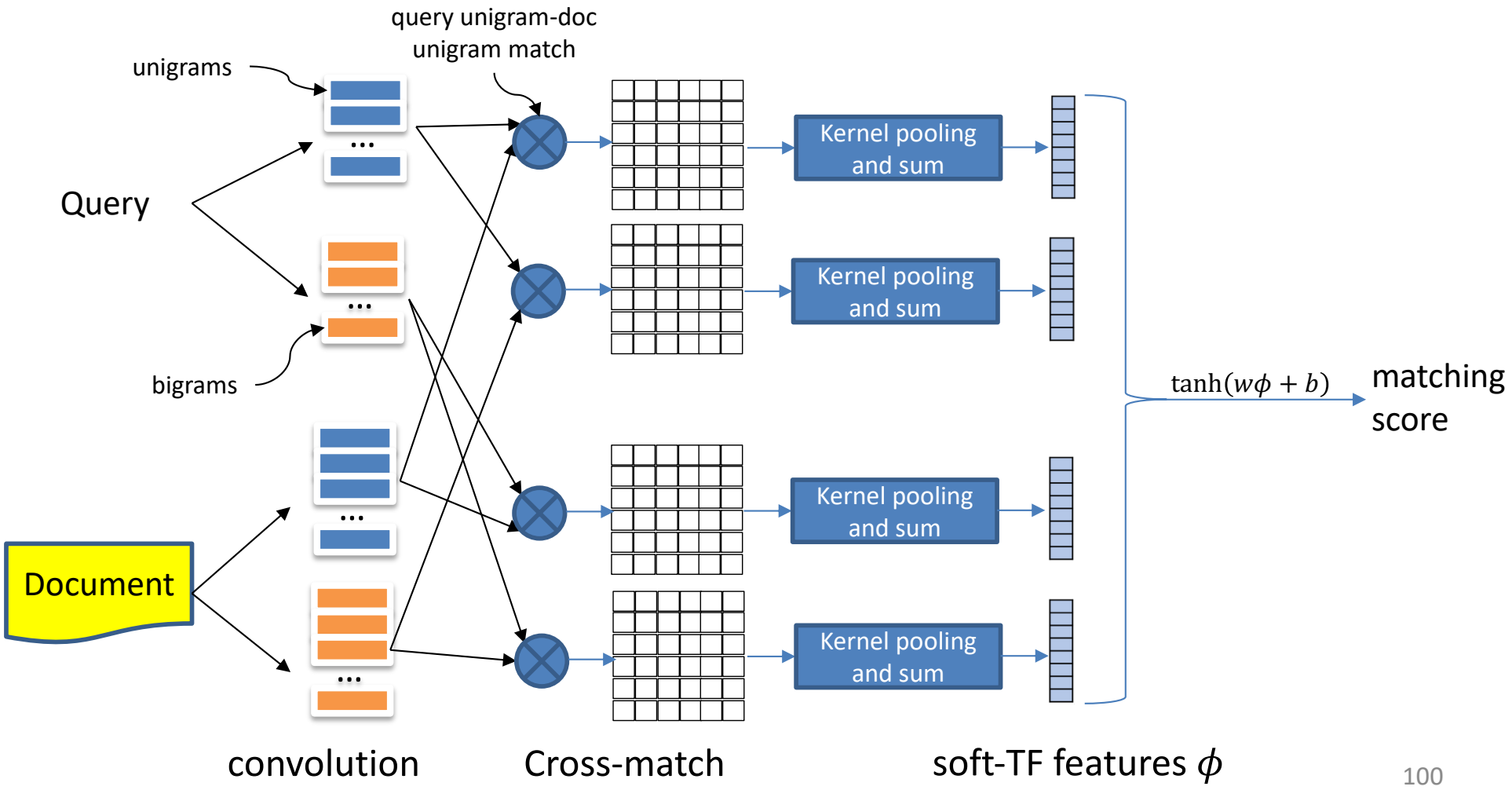
K-NRM: Kernel Pooling as Matching Function (Xiong et al., SIGIR '17)

- Basic matching signals: cosine similarity of word embeddings
- Ranking function: kernel pooling + nonlinear feature combination
- Semantic gap: embedding bridge the semantic gap
- Word order: kernel pooling and sum operations lost order information



Conv-KNRM (Dai et al., WSDM '18)

- Based on KNRM
- N-gram cross-matching to capture local word order information



Experimental Evaluation

	Method	NDCG@1	NDCG@10
Traditional IR / Learning to rank	BM25	0.142	0.287
	RankSVM	0.146	0.309
Representation Learning Methods	CDSSM	0.144	0.333
Matching Function Learning	MatchPyramid	0.218	0.379
Query-Document Relevance Matching	DRMM	0.137	0.315
	K-NRM	0.264	0.428
	Conv-KNRM	**0.336**	**0.481**

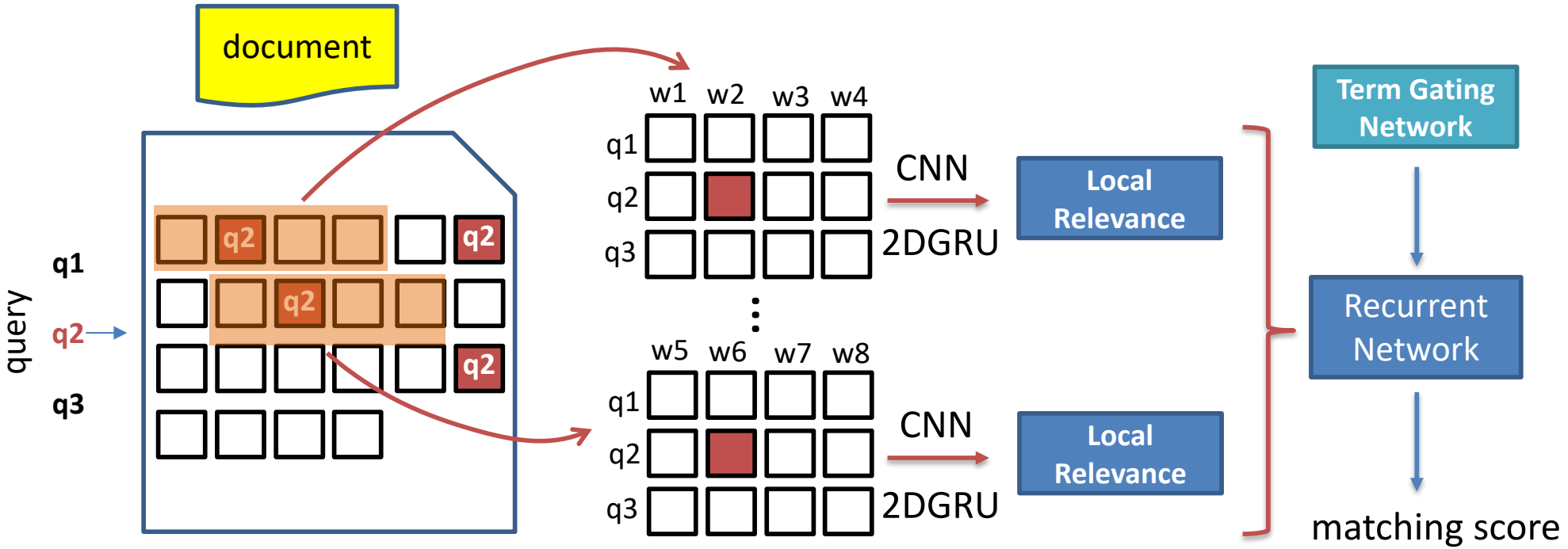
Results reported in (Dai et al., WSDM '18), based on Sogou Log dataset (95,229 queries)

Relevance Matching based on Local Context of Matched Terms

- Step 1: for each query term
  - Identify its local contexts (area around exact matching positions) among the document
  - Conduct matching between query and local contexts
- Step 2: aggregate the local matching signals
- Advantages:
  - Matching between short query and long document text
  - Robust: filter out irrelevant document contents
  - Keep order information within each local context

DeepRank (Pang et al., CIKM '17)

- Calculate relevance by mimicking the human relevance judgement process



1. **Detecting Relevance locations**: focusing on locations of query terms when scanning the whole document

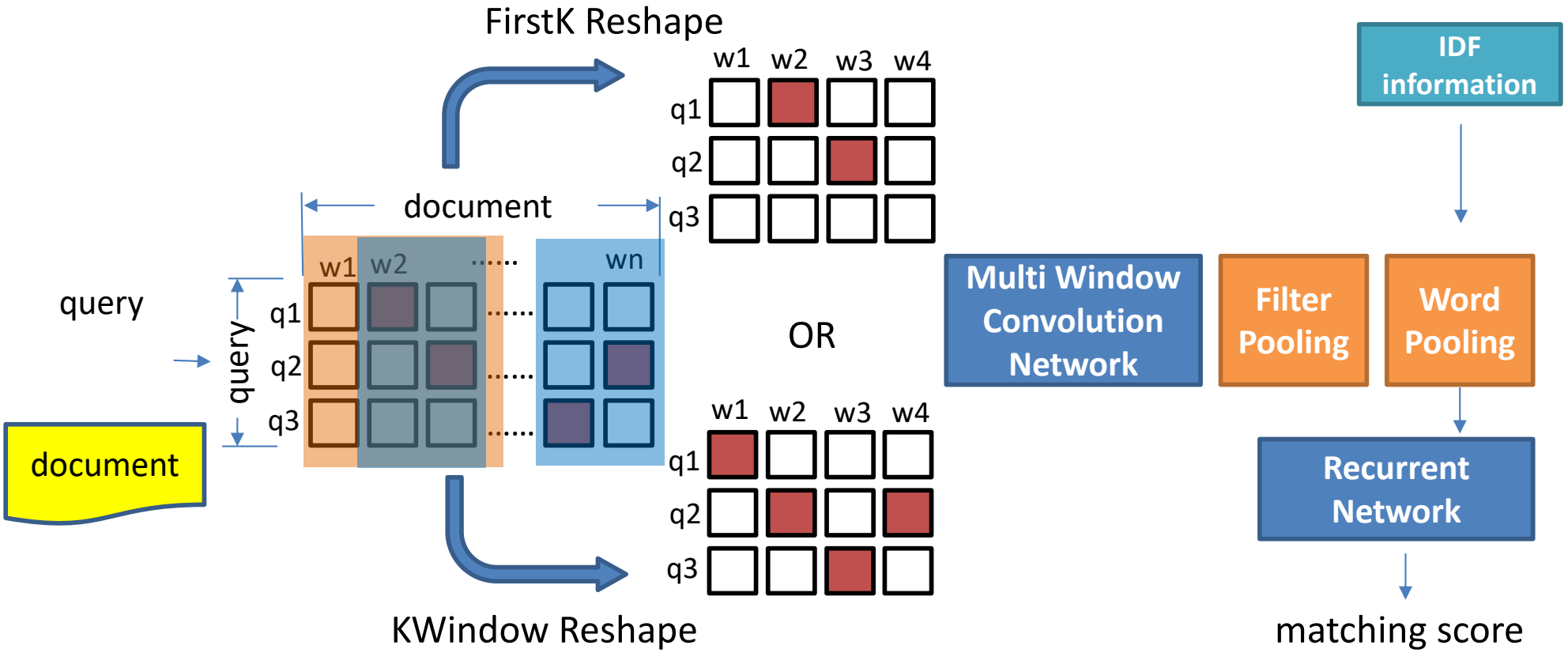
2. **Determining local relevance**: relevance between query and each location context, using MatchPyramid / MatchSRNN etc.

3. **Matching signals aggregation**:

$$F(\mathbf{q}, \mathbf{d}) = \sum_{w \in \mathbf{q}} (E_w \mathbb{I})^T \cdot \mathcal{T}(w)$$

Position-Aware Neural IR Model (PACRR, Hui et al., EMNLP '17)

- Hypothesis: relevance matching is determined by some positions in documents
  - First k words in document (FirstK), or
  - Most similar context positions in document (Kwindow)



Experimental Evaluation

	Method	NDCG@20	ERR@20
Traditional IR	QL	0.231	0.131
Matching Function Learning	MatchPyramid	0.278	0.176
Global Distribution of Matching Signals	DRMM	0.300	0.193
	K-NRM	0.324	0.201
	DUET	0.267	0.179
Local Context of Matched Terms	PACRR-firstk	0.339	0.221

Results reported in (Hui et al., EMNLP '17), based on Web Track 14 dataset.

Short Summary

- Methods based on global distributions of matching strengths
  - 1. calculating term matching strength distributions
  - 2. aggregating the distributions to a matching score
- Methods based on local context of matched terms
  - 1. Identifying the relevance locations / contexts
  - 2. Matching the whole query with the local contexts
  - 3. Aggregating the local matching signals

References

- Clark J. Google turning its lucrative web search over to ai machines[J]. Bloomberg Technology. Publicado em, 2015, 26.
- Metz C. AI is transforming Google search[J]. The rest of the web is next. WIRED Magazine, 2016.
- Huang P S, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data[C]//Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013: 2333-2338.
- Hu B, LuShen Y, He X, Gao J, et al. A latent semantic model with convolutional-pooling structure for information retrieval[C]//Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. ACM, 2014: 101-110.
- Z, Li H, et al. Convolutional neural network architectures for matching natural language sentences[C]//Advances in neural information processing systems. 2014: 2042-2050.
- Qiu X, Huang X. Convolutional Neural Tensor Network Architecture for Community-Based Question Answering[C]//IJCAI. 2015: 1305-1311.
- Palangi H, Deng L, Shen Y, et al. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval[J]. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2016, 24(4): 694-707.
- Yin W, Schütze H. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015, 1: 63-73.
- Socher R, Huang E H, Pennin J, et al. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection[C]//Advances in neural information processing systems. 2011: 801-809.
- Wan S, Lan Y, Guo J, et al. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations[C]//AAAI. 2016, 16: 2835-2841.

References

- Pang L, Lan Y, Guo J, et al. Text Matching as Image Recognition[C]//AAAI. 2016: 2793-2799.
- Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-SRNN: modeling the recursive matching structure with spatial RNN. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), 2922-2928.
- Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In Proceedings of EMNLP, 2016.
- Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In Proceedings of WSDM 2018.
- Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, Russell Power. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In Proceedings of SIGIR 2017.
- Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In Proceedings of WWW 2017.
- Jiafeng Guo, Yixing Fan, Qiqing Yao, W. Bruce Croft, A Deep Relevance Matching Model for Ad-hoc Retrieval. In Proceedings of CIKM 2016.
- Liu Yang, Qingyao Ai, Jiafeng Guo, W. Bruce Croft, aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In Proceedings of CIKM 2016.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu and Xueqi Cheng. DeepRank: a New Deep Architecture for Relevance Ranking in Information Retrieval. In Proceedings of CIKM 2017.
- Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He. CA-RNN: Using Context-Aligned Recurrent Neural Networks for Modeling Sentence Similarity. . In Proceedings of AAAI 2018.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Xueqi Cheng. A Study of MatchPyramid Models on Ad-hoc Retrieval. In Proceedings of SIGIR 2016 Neu-IR Workshop.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. WSDM '18, 279-287.

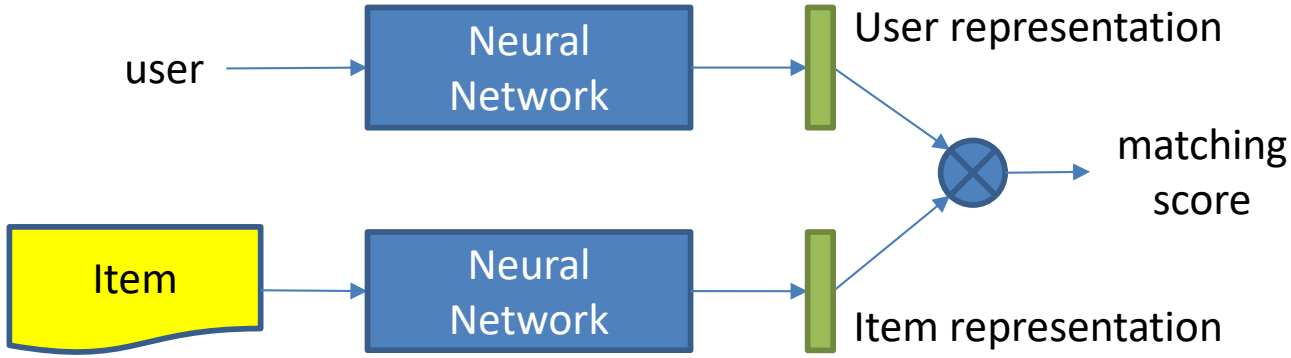
Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
  - Deep matching models for search
  - Deep matching models for recommendation
- Summary

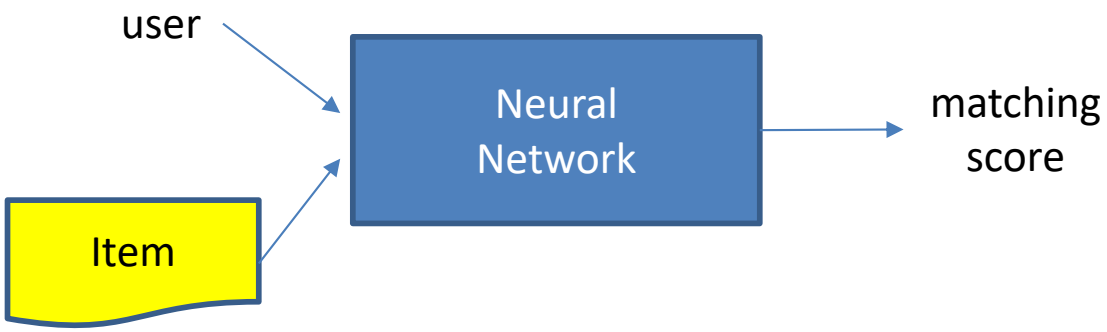
Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Deep Matching Models for Recommendation

- Methods of representation learning



- Methods of matching function learning



Methods of Representation Learning

1. Collaborative Filtering:

   Models are built based on user-item interaction matrix only.

   - **DeepMF**: Deep Matrix Factorization (Xue et al, IJCAI'17)
   - **AutoRec**: Autoencoders Meeting CF (Sedhain et al, WWW'15)
   - **CDAE**: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)

2. Collaborative Filtering + Side Info:

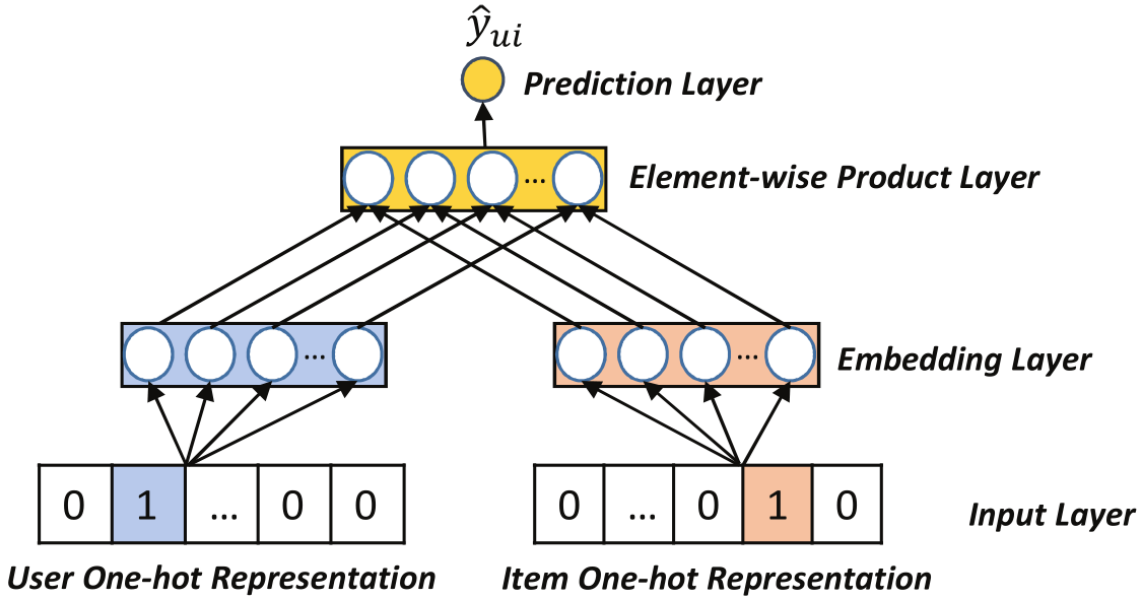
   Models are built based on user-item interaction + side info.

   - **DCF**: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)
   - **DUIF**: Deep User-Image Feature (Geng et al, ICCV'15)
   - **ACF**: Attentive Collaborative Filtering (Chen et al, SIGIR'17)
   - **CKB**: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)

Recap MF as a Neural Network

- **Input**: user -> ID (one-hot), item -> ID (one-hot).
- **Representation Function**: linear embedding layer.
- **Matching Function**: inner product.

$$f_{MF}(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^\top \mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik},$$



Deep Matrix Factorization (Xue et al, IJCAI'17)

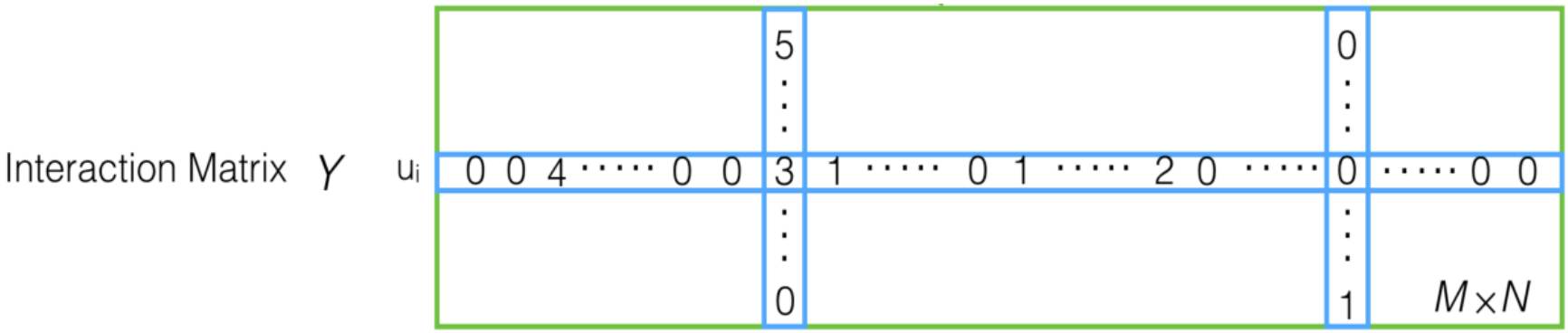
- **Input**:

  user -> items that she has rated (multi-hot), i.e., row vector of Y

  indicates the user's global preference

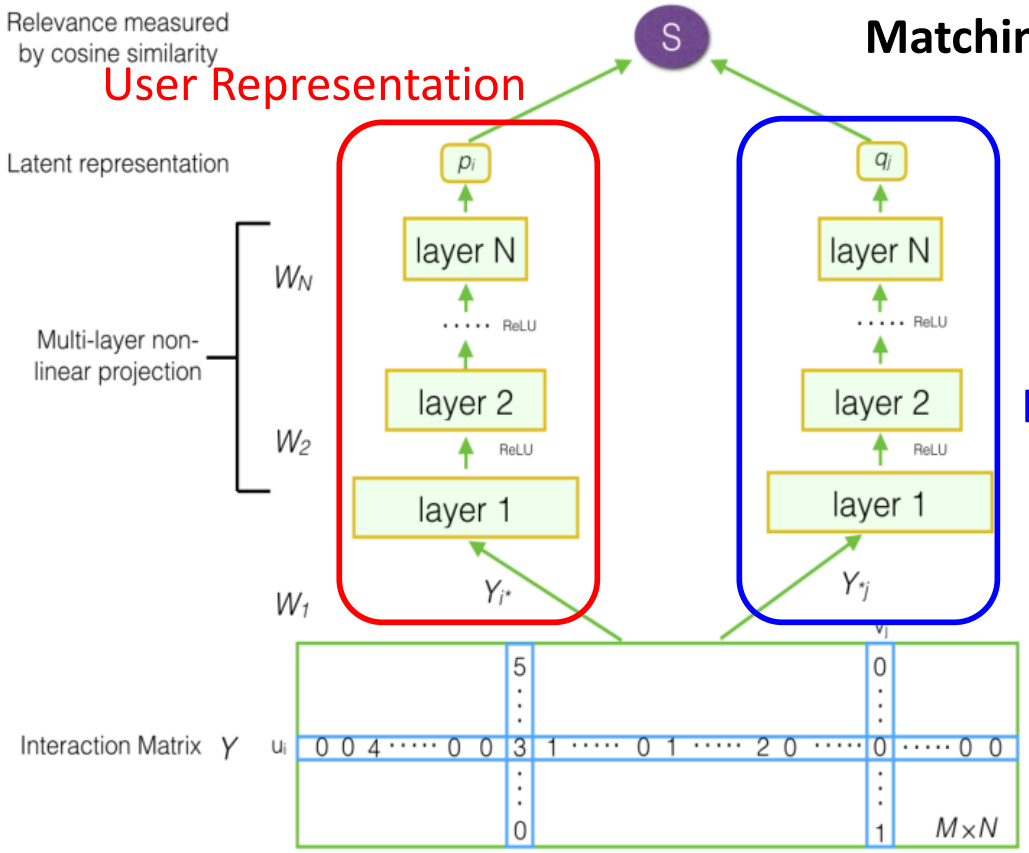
  item -> users who have rated it (multi-hot), i.e., column vector of Y

  indicates the item's rating profile.



Deep Matrix Factorization (Xue et al, IJCAI'17)

- **Representation Function**:
  - Multi-Layer Perceptron



Matching Function: cosine similarity

$$cosine(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \, \|q_j\|}$$

Item Representation

AutoRec (Sedhain et al, WWW'15)

- **Input: (similar to DeepMF)**

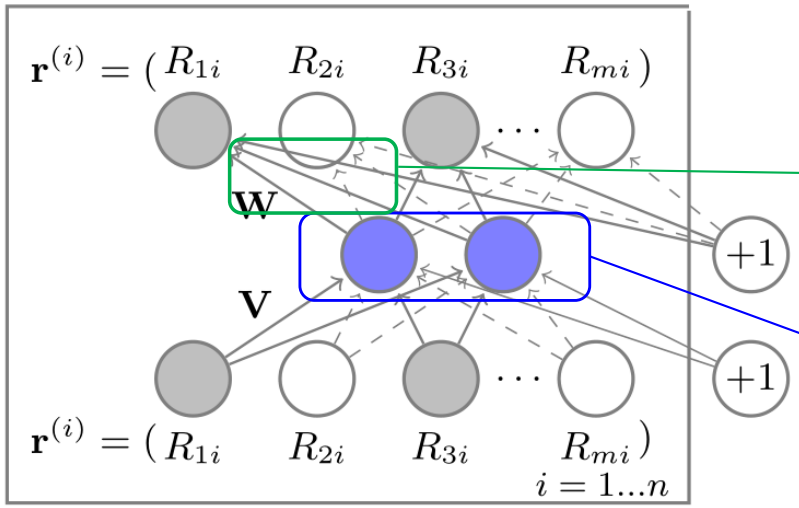
  user -> historically rated items (user-based autoencoder).

  item-> ID

- **Representation Function:** Multi-Layer Perceptron
- **Matching Function**: inner product

Input reconstruction: $h(\mathbf{r};\theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$

$$\min_{\theta} \sum_{i=1}^{n} ||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)};\theta))||_{\mathcal{O}}^2 + \frac{\lambda}{2} \cdot (||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2),$$



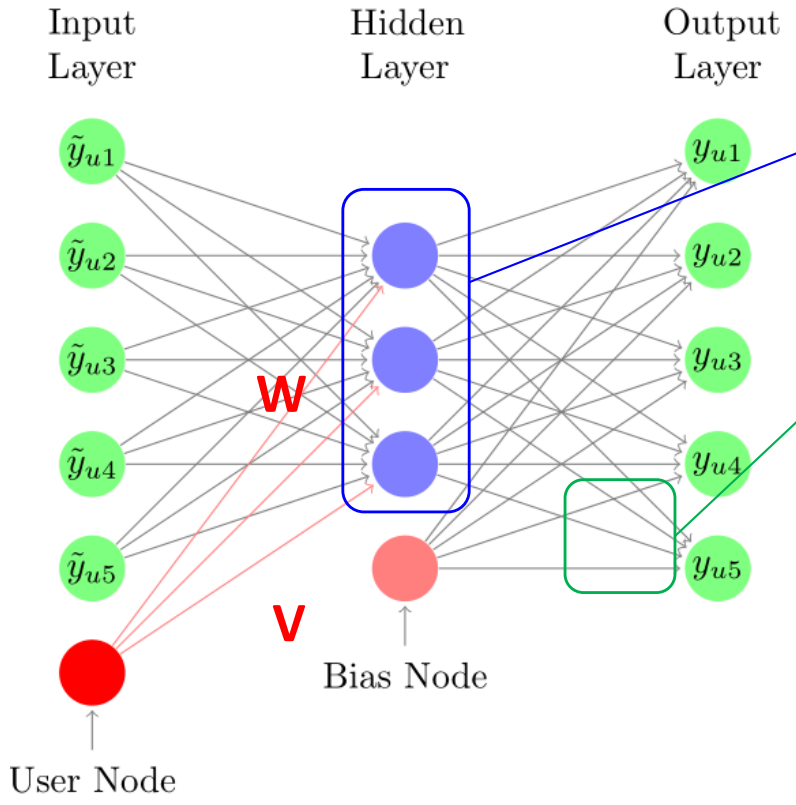
Output weights denote item representation

Hidden neurons denote user representation

user-based autoencoder

Collaborative Denoising Auto-Encoder (Wu et al, WSDM'16)

- **Input:**
  user -> ID & historically rated items (similar to SVD++)
  item -> ID
- **Representation Function**: Multi-Layer Perceptron



Hidden neurons are user representation:

$$\boldsymbol{z}_u = h\left(\boldsymbol{W}^\top \tilde{\boldsymbol{y}}_u + \boldsymbol{V}_u + \boldsymbol{b}\right),$$

Weights of output layer are item representation

$$\hat{y}_{ui} = f\left(\boldsymbol{W}_i'^\top \boldsymbol{z}_u + b_i'\right),$$

Short Summary

- Either ID or history is used as the profile of user/item
- Models with history as input are more expressive, but are also more expensive to train.

- The Auto-Encoder architecture is essentially identical to

  MLP (representation learning) + MF (matching function).

  Nonlinear Linear

Methods of Representation Learning

1. Collaborative Filtering:

   Models are built based on user-item interaction matrix only.

   - **DeepMF**: Deep Matrix Factorization (Xue et al, IJCAI'17)
   - **AutoRec**: Autoencoders Meeting CF (Sedhain et al, WWW'15)
   - **CDAE**: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)

2. Collaborative Filtering + side info:

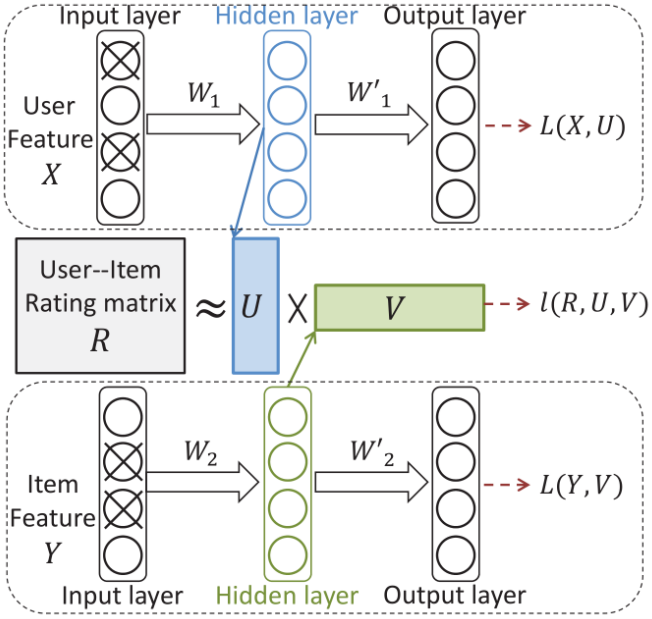
   Models are built based on user-item interaction matrix + side info.

   - **DCF**: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)
   - **DUIF**: Deep User-Image Feature (Geng et al, ICCV'15)
   - **ACF**: Attentive Collaborative Filtering (Chen et al, SIGIR'17)
   - **CKB**: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)

Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)

- Denoising Auto-Encoder is used to learn features (hidden layers) of user and item from side information.
- The predictive model is MF.

User age, gender, city, occupation, locations …



Item genres, title, texts

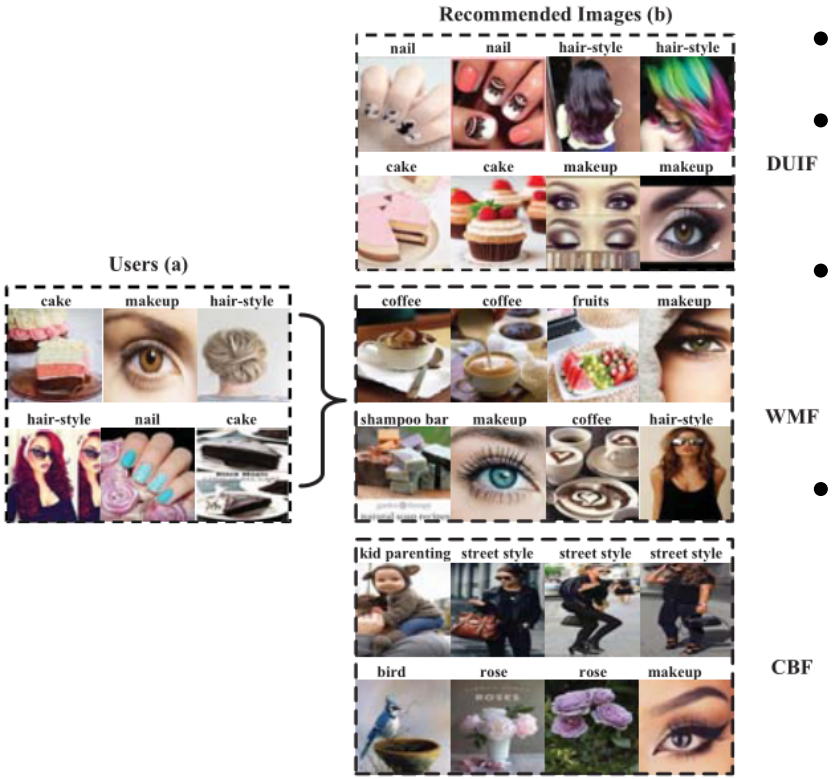
$$\arg \min_{U,V,W_1,W_2,P_1,P_2} \mathcal{L}_U(W_1,P_1,U) + \mathcal{L}_V(W_2,P_2,V) + \alpha\|A \odot (R - UV^\top)\|_F^2 + \beta(\|U\|_F^2 + \|V\|_F^2)$$

User features reconstruction

Item features reconstruction

Matrix Factorization

DUIF: Deep User and Image Feature Learning (Geng et al, ICCV'15)



- Task: collaborative image recommendation
- Deep CNN (AlexNet) is used to extract features for images
- The deep image features (dim=4096) are projected to user latent space (dim=300) by using linear projection.
- The predictive model is MF:

$$\hat{y}_{ui} =< \mathbf{p}_u, \mathbf{W}^T \text{CNN}(\mathbf{f}_i) >,$$

Linear Projection — Image raw features

- The overall model (MF+W+CNN) is trained end-to-end.

ACF: Attentive Collaborative Filtering (Chen et al, SIGIR'17)

- **Input:**

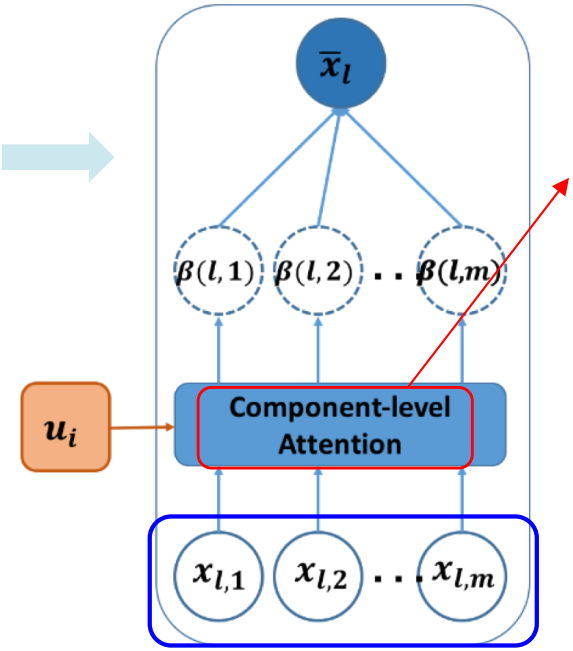
  user -> ID & historical interacted items.

  Item -> ID & visual features.

- **Item Representation**:

  Component-level attention -> components contribute differently to an item's representation

A user's preference on different components of the item are not equal!



$$\bar{\mathbf{x}}_l = \sum_{m=1}^{|\{\mathbf{x}_{l*}\}|} \beta(i,l,m) \cdot \mathbf{x}_{lm}$$

Attention Net determines each region's weight:
- Input: user embedding and region features

Features of each region

ACF: Attentive Collaborative Filtering (Chen et al, SIGIR'17)

- **Input:**

  user -> ID & historical interacted items.

  item -> ID & visual features.

- **User Presentation:**
  - Item-level attention -> historical items contribute differently to a Attentive SVD++ model:

$$\hat{R}_{ij} = \left(\mathbf{u}_i + \sum_{l \in \mathcal{R}(i)} \alpha(i,l) \mathbf{p}_l \right)^T \mathbf{v}_j$$

A user's preference on different items of user history are not equal!

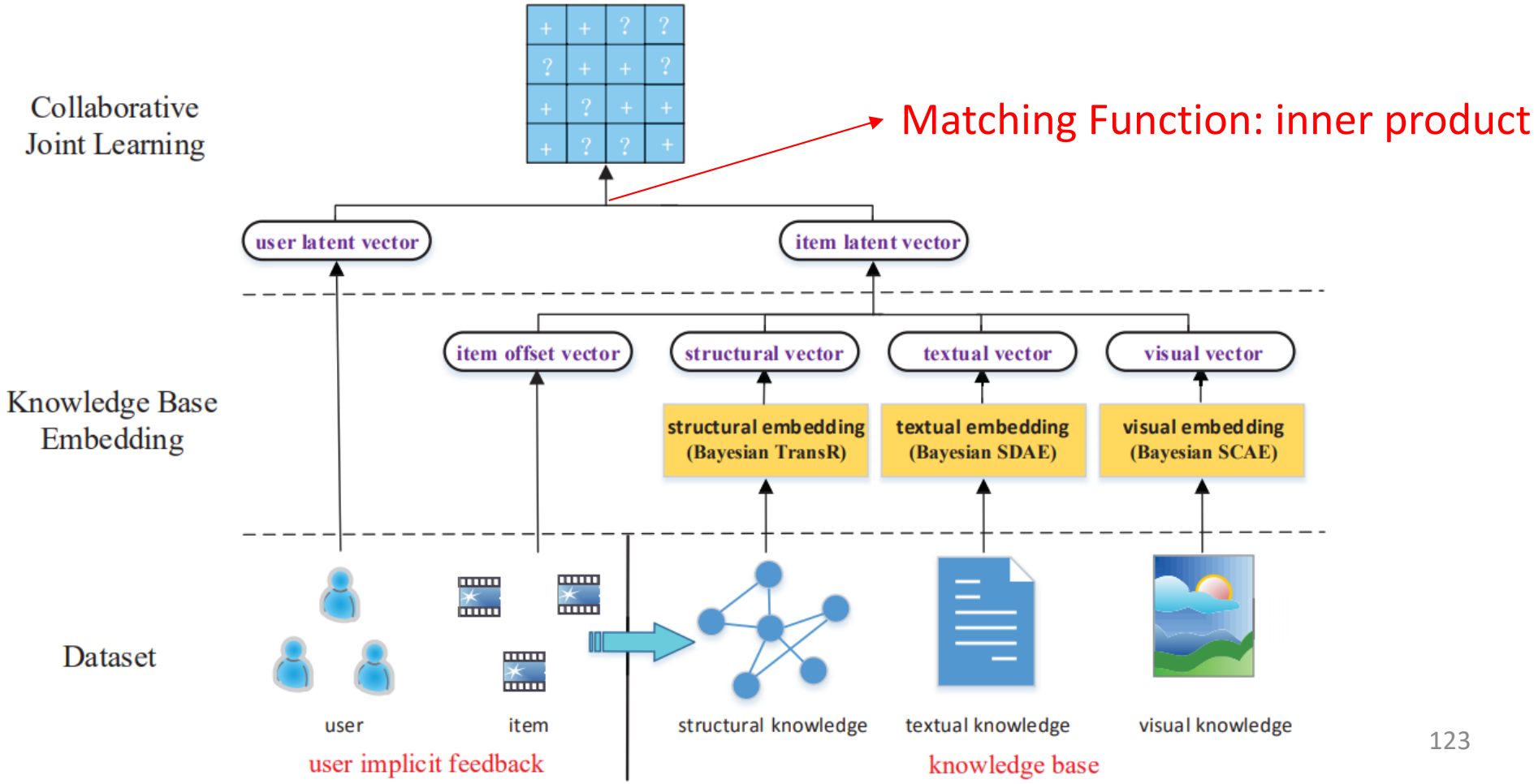
Attention weight determines each item's weight:
- Input: user embedding and item embedding

CKE: Collaborative Knowledge Base Embedding (Zhang et al, KDD'16)

- **Input:**

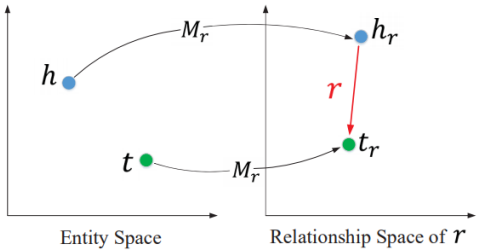
  user -> ID

  item -> ID + Information in KB (structural, textual, visual)



CKE: Collaborative Knowledge Base Embedding (Zhang et al, KDD'16)

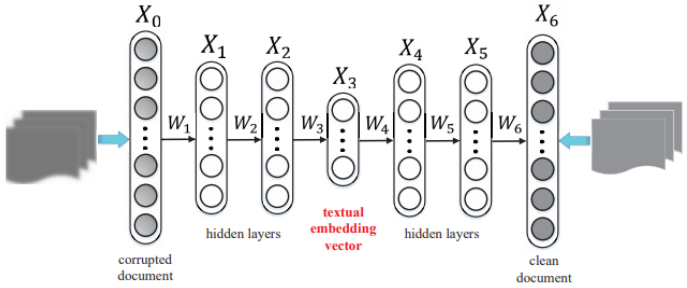
- **Representations:**
  - Structural embedding: TransR, TransE, …



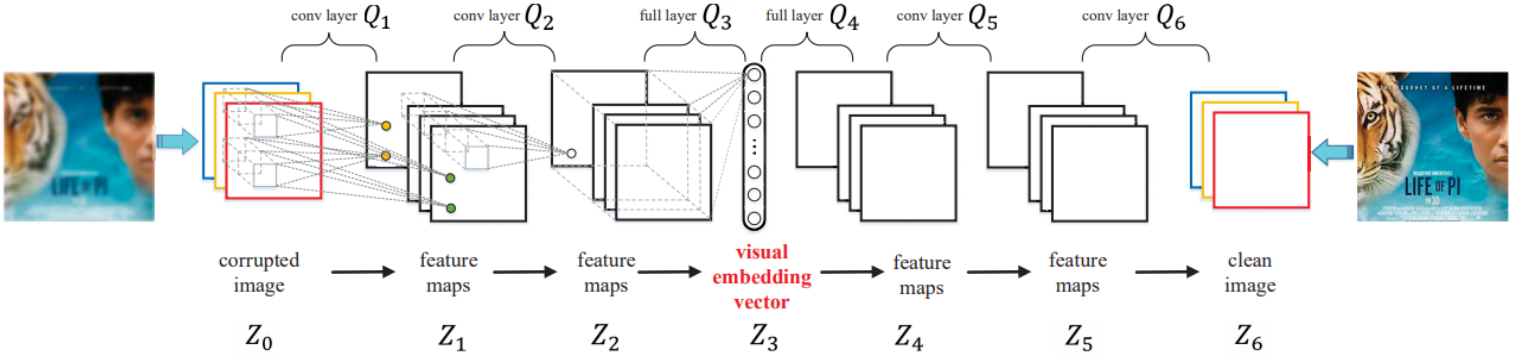
$$\mathbf{v}_h^r = \mathbf{v}_h \mathbf{M}_r, \qquad \mathbf{v}_t^r = \mathbf{v}_t \mathbf{M}_r.$$

$$f_r(v_h, v_t) = ||\mathbf{v}_h^r + \mathbf{r} - \mathbf{v}_t^r||_2^2.$$

  - Textual embedding: stacked denoising auto-encoders (S-DAE)

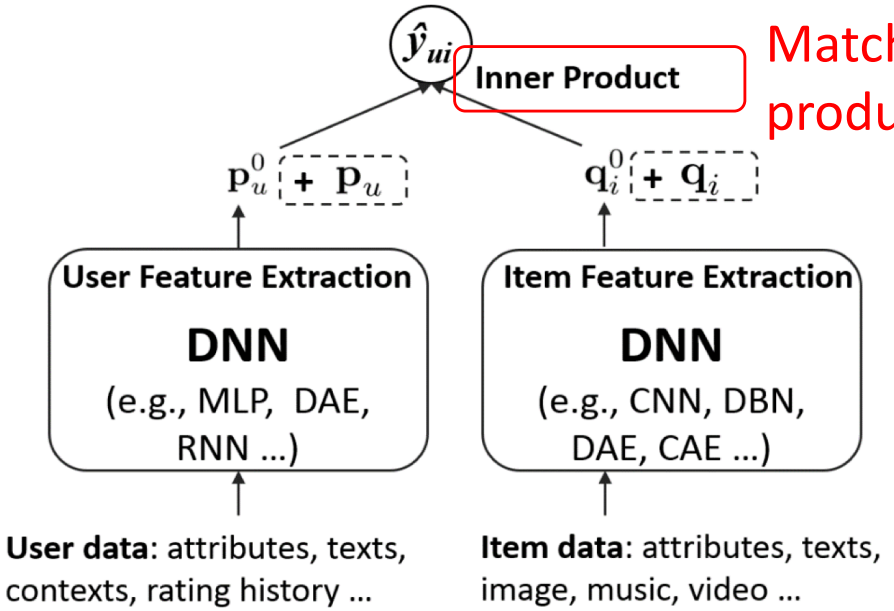


  - Visual embedding: stacked convolutional auto-encoders (SCAE)



Short Summary

- A General framework to summarize the above works:



Matching function is a simple inner product or cosine similarity

- Depending on the available data to describe a user/item, we can choose appropriate DNN to learn representation.

  E.g., Textual Attributes -> AutoRec, Image -> CNN, Video -> RNN etc.

Next: Methods of Matching Function Learning

1. CF models:
   - Based on Neural Collaborative Filtering (NCF) framework:
     NeuMF: Neural Matrix Factorization (He et al, WWW'17)
     ConvNCF: Outer Product-based NCF (He et al, IJCAI'18)
   - Based on Translation framework:
     TransRec: Translation-based Recommendation (He et al, Recsys'17)
     LRML: Latent Relational Metric Learning (Tay et al, WWW'18)
2. Feature-based models:
   - Based on Multi-Layer Perceptron:
     Wide&Deep (Cheng et al, DLRS'16),
     Deep Crossing (Shan et al, KDD'16)
   - Based on Factorization Machines (FM):
     Neural FM (He and Chua, SIGIR'17),
     Attentional FM (Xiao et al, IJCAI'17),
     DeepFM (Guo et al, IJCAI'17)

Neural Collaborative Filtering Framework (He et al, WWW'17)

- **NCF is a general framework that replaces the inner product with a neural network to learn the matching function.** $\hat{y}_{ui} = f(\mathbf{p}_u, \mathbf{q}_i)$

Matching function based on NN

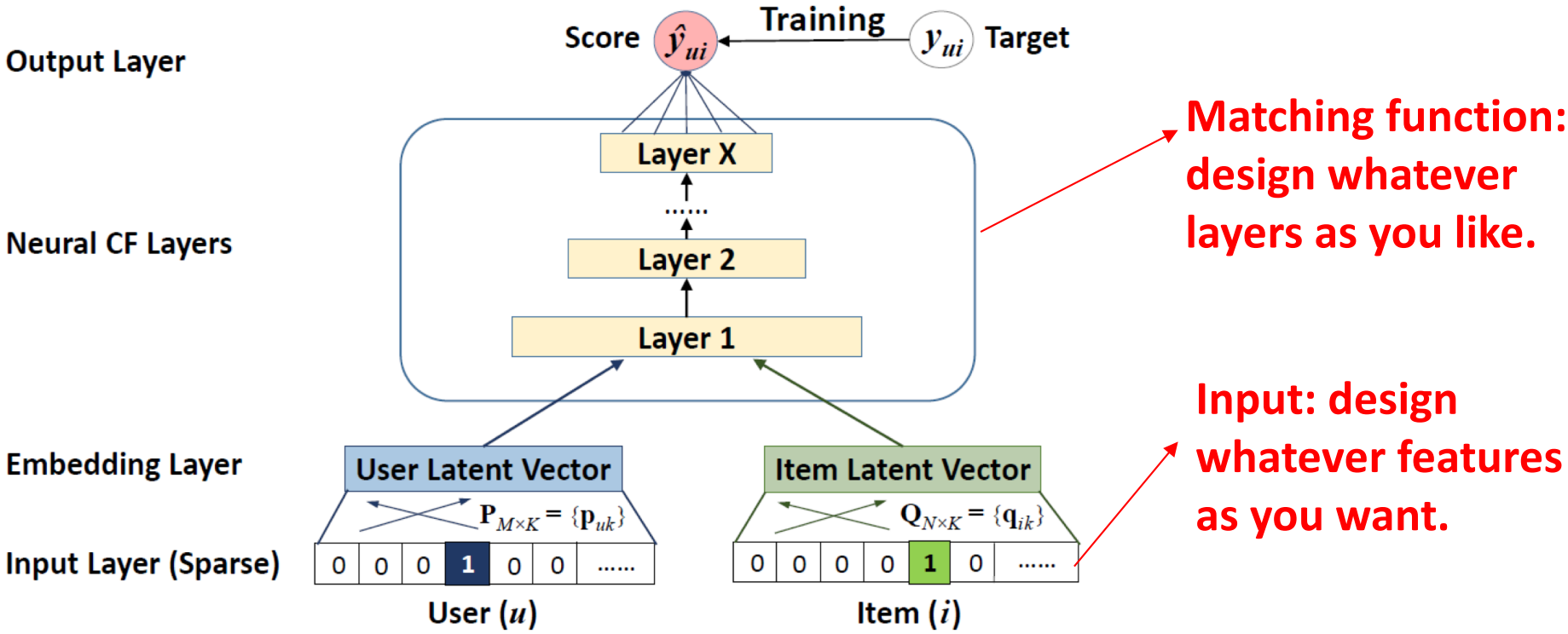
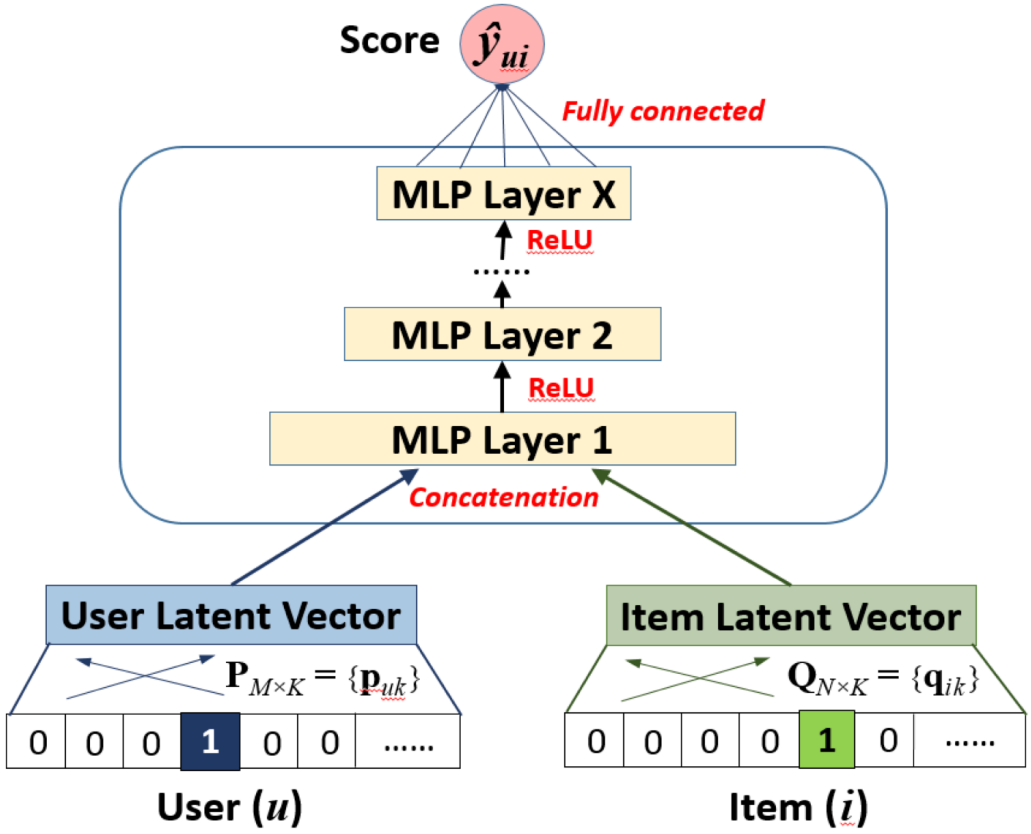


Figure 2: Neural collaborative filtering framework

Multi-Layer Perceptron for CF

- The most intuitive idea is to use Multi-Layer Perceptron as the matching function.



Unfortunately, MLP doesn't perform well and underperforms MF.

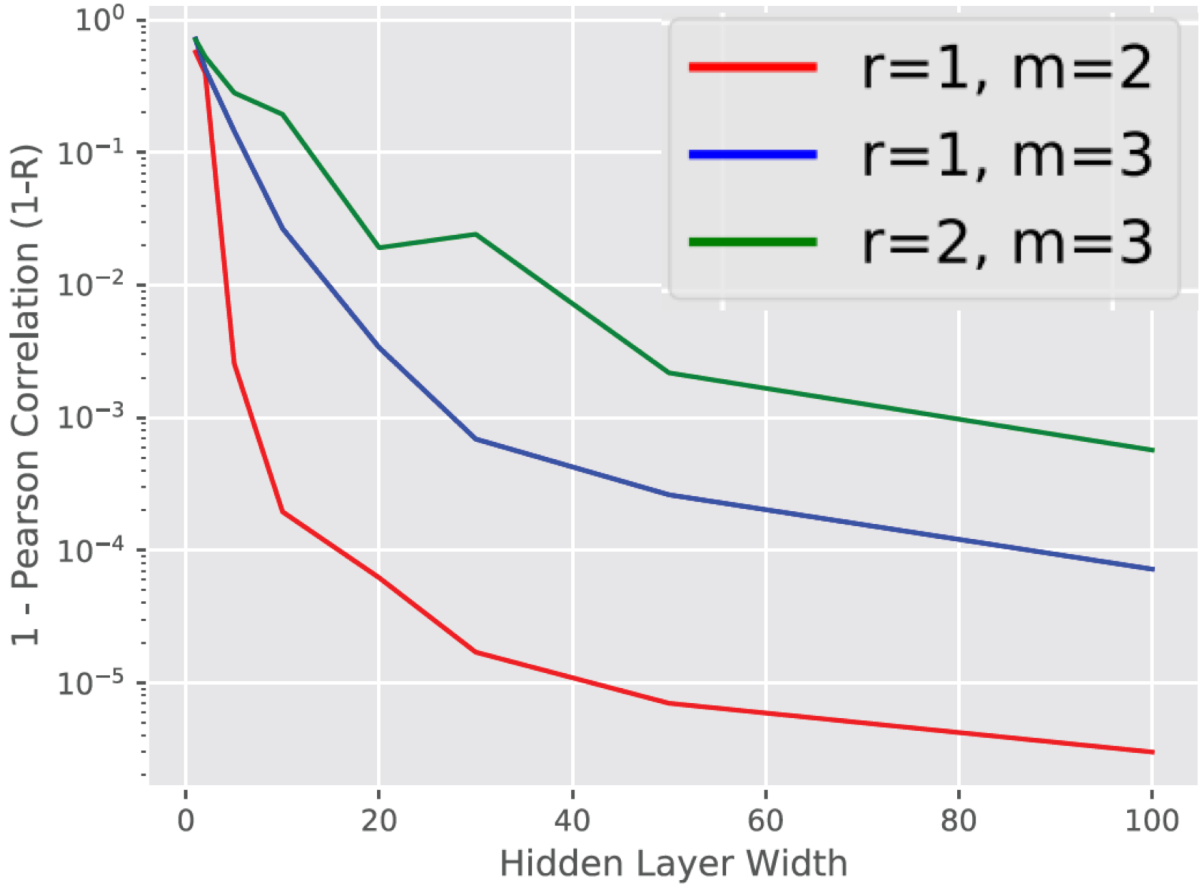
Why?

(He et al, WWW'17)

DNN is Weak in Capturing Multiplicative Relation

- Evidence from Google researchers (Beutel et al, WSDM'18)
  - Setting: generate low-rank data, and use one-layer MLP to fit it

r: rank size; m: data dimension (2 -> matrix; 3 -> 3D tensor).



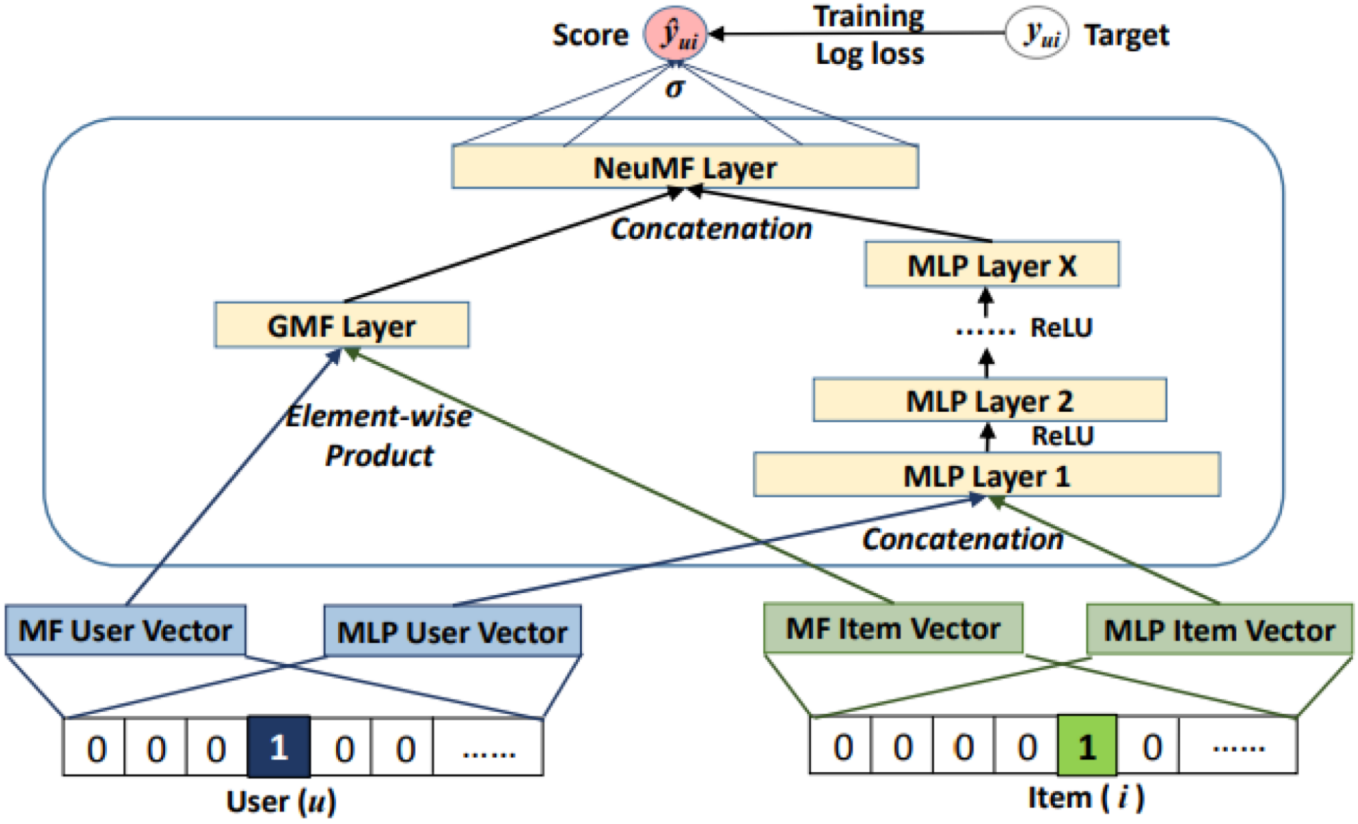
MLP can learn low-rank relation, but is inefficient in doing so!

- Need to use 100 neurons to fit a rank-1 matrix.

Insight: need to augment DNN with multiplicative relation modeling!

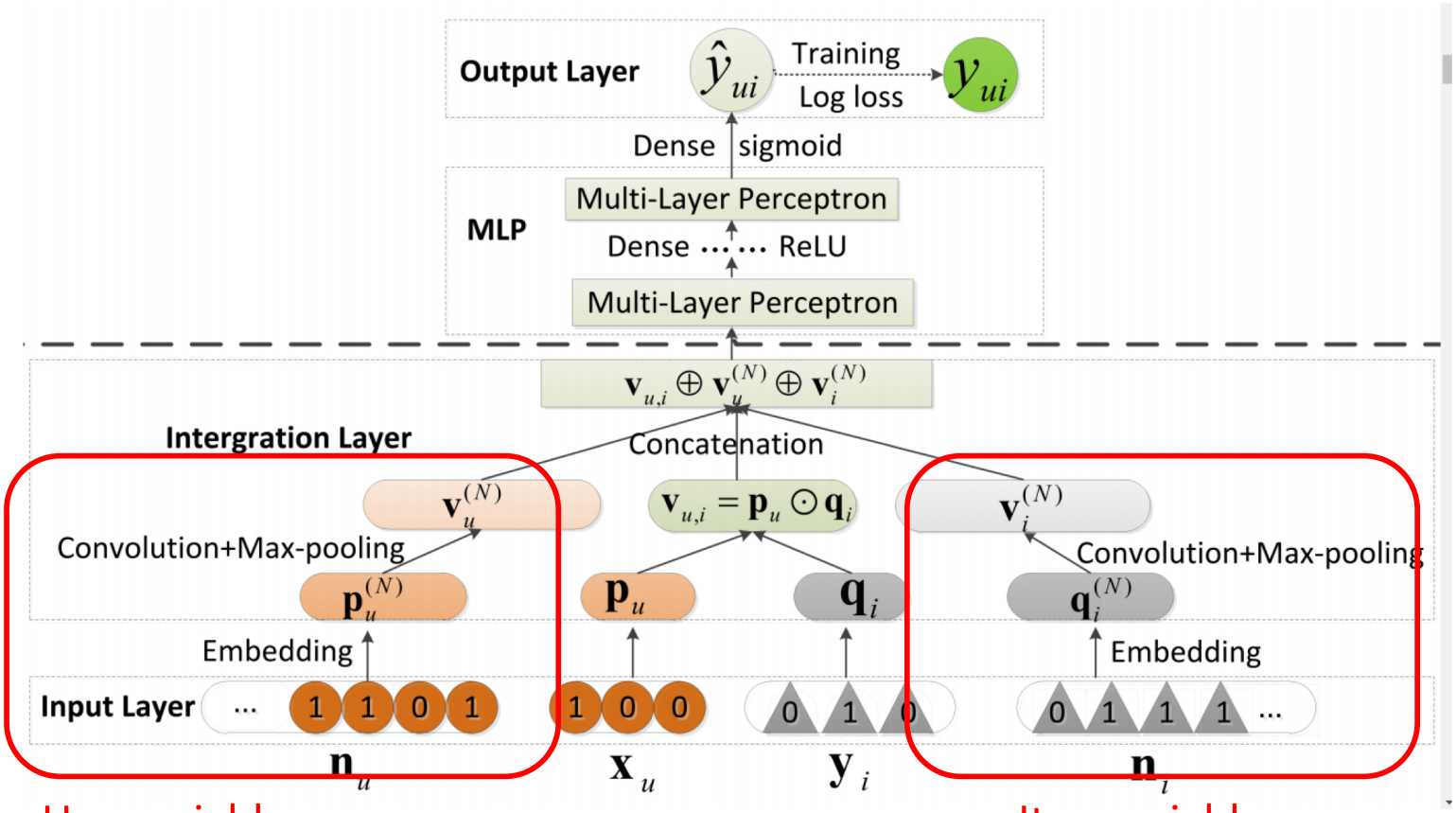
NeuMF: Neural Matrix Factorization (He et al, WWW'17)

- NeuMF unifies the strengths of MF and MLP in learning the matching function:
  - MF uses inner product to capture the multiplicative relation
  - MLP is more flexible in using DNN to learn the matching function.



NNCF: Neighbor-based NCF (Bai et al, CIKM'17)

- Feeding user and item neighbors into the NCF framework
  - Direct neighbors or indirect community neighbors are considered.



Experiment Results(Bai et al, CIKM'17)

Datasets	#Interaction	# Users	#Items	Sparsity
Delicious	437,593	1,867	69,223	99.66%
MovieLens	1,000,209	3,706	6,040	95.53%

Performance Comparison on Item Recommendation (%)

Datasets	Delicious		MovieLens	
Models	HR@5	NDCG@5	HR@5	NDCG@5
ItemPop	5.41	3.22	31.49	20.18
ItemKNN	59.69	55.90	45.01	30.14
MF-BPR	73.77	74.11	51.03	36.21
NeuMF	85.53	80.68	56.55	38.30
NNCF	87.31	84.58	62.00	42.21

CF method is better than non-personalized method

Model-based CF is better than memory-based CF

Deep NCF models are better than shallow MF models by a large margin.

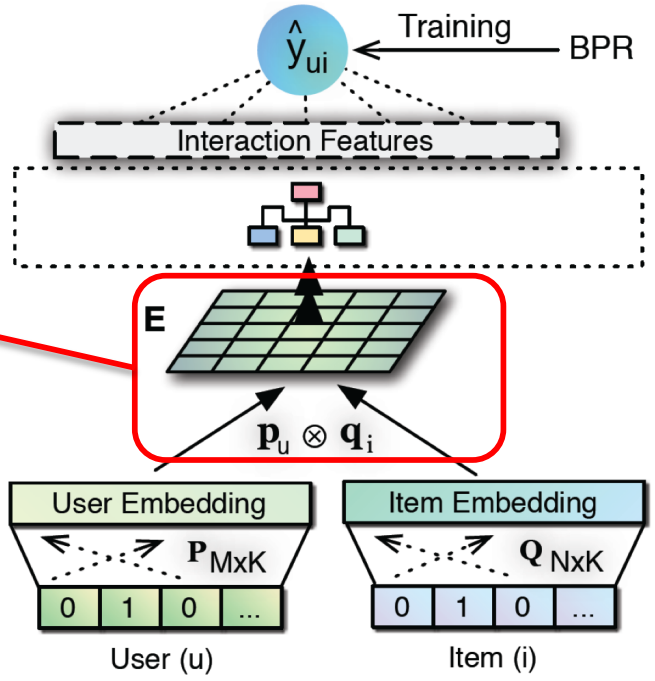
ONCF: Outer-Product based NCF (He et al, IJCAI'18)

- The above NCF models merge user embedding and item embedding with element-wise product or concatenation:
  - Implicitly assume that embedding dimensions are independent.
- How to model the relations between embedding dimensions?
- ONCF applies outer-product on user embedding and item embedding:
  - Explicitly models pairwise correlations between embedding dimensions:

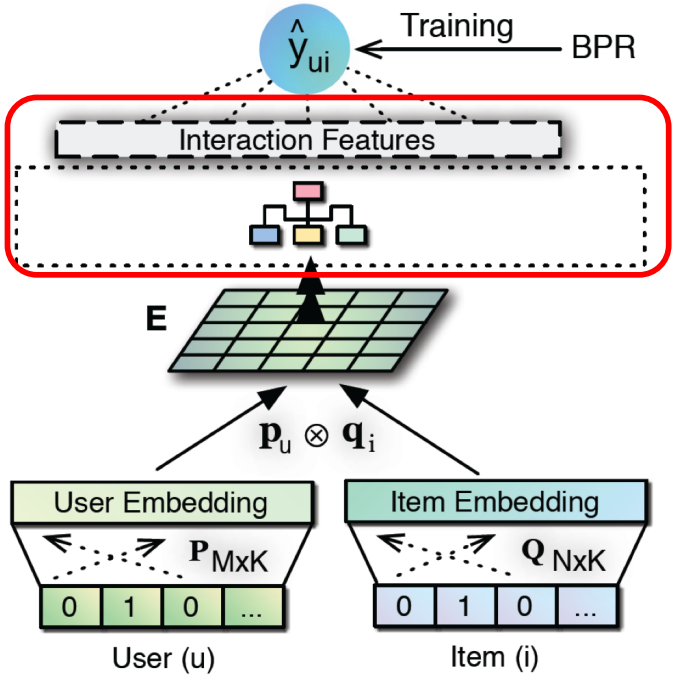
Outer-product gets a 2D "interaction map":

$$e_{k_1,k_2} = p_{u,k_1} q_{i,k_2}$$

- Diagonal elements are inner product



ONCF: Outer-Product based NCF (He et al, IJCAI'18)



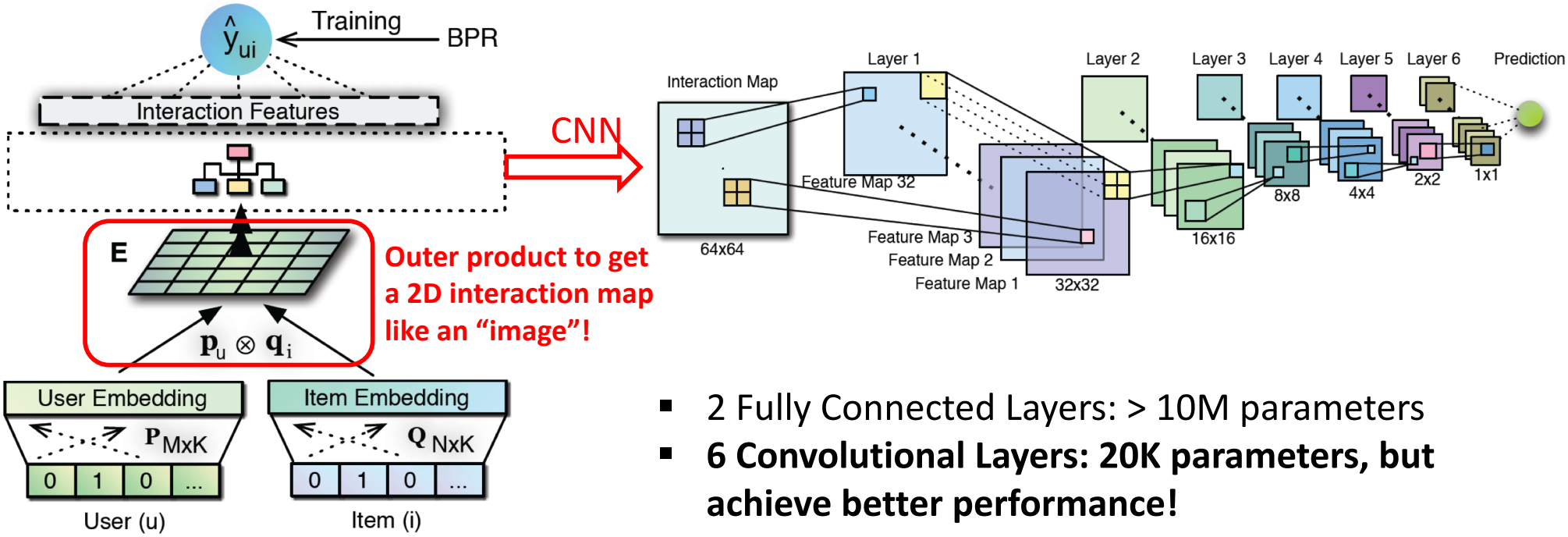
Above the interaction map are hidden layers, which aim to extract useful signal from the 2D interaction map.

A straightforward solution is to use MLP, however it results in too many parameters:

- Interaction map **E** has K × K neurons (K is embeddings size usually hundreds)
- Require **large memories** to store the model
- Require **large training data** to learn the model well

Convolutional NCF (ConvNCF) (He et al, IJCAI'18)

- ConvNCF uses locally connected CNN as hidden layers in ONCF:
  - CNN has much fewer parameters than MLP
  - Hierarchical tower structure: higher layer integrates more information from larger area.
  - Final prediction summarizes all information from interaction map.



- 2 Fully Connected Layers: > 10M parameters
- **6 Convolutional Layers: 20K parameters, but achieve better performance!**

Experiment Results (He et al, IJCAI'18)

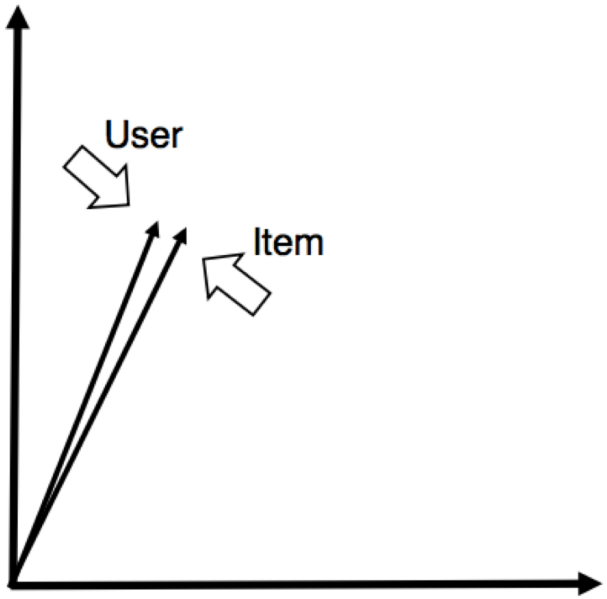
Datasets	#Interactions	#Users	#Items	Sparsity
Yelp	730,791	25,815	25,677	99.89%
Gowalla	1,249,703	54,156	52,400	99.95%

Datasets	Gowalla		Yelp	
Models	HR@5	NDCG@5	HR@5	NDCG@5
ItemPop	20.03	10.99	7.10	3.65
MF-BPR	62.84	48.25	17.52	11.04
MLP	63.59	48.02	17.66	11.03
IRGAN	63.89	49.58	18.61	11.98
NeuMF	67.44	53.19	18.81	11.89
ConvNCF	69.14	54.94	19.06	12.09

ConvNCF are better than NeuMF and MLP with much fewer parameters.

Overview of Translation-based Models (Tay et al, WWW'18)

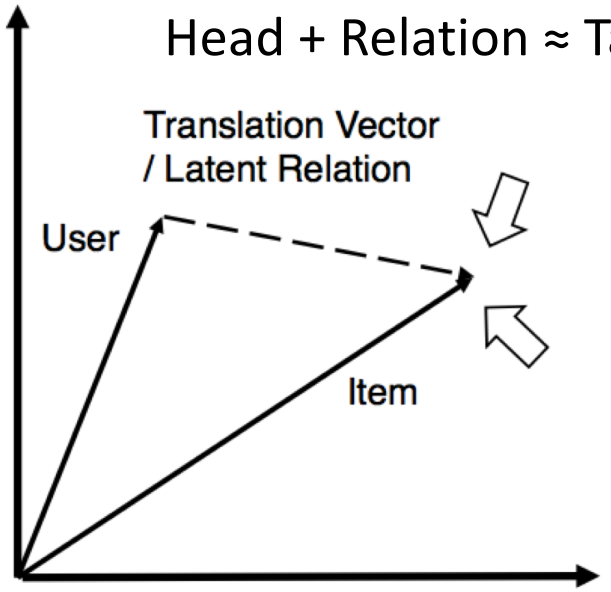
MF-based model:



- Push ***user vector*** close to ***item vector***
- Measure closeness by inner product (or cosine similarity)

Translation-based:

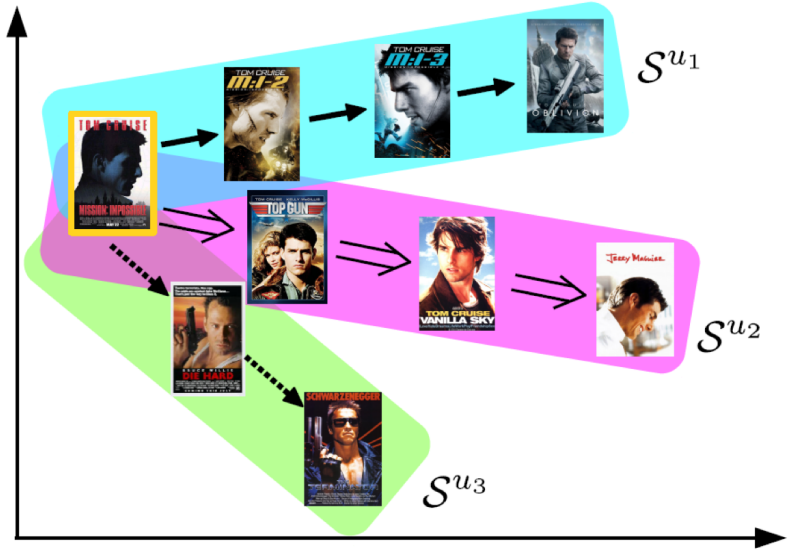
Head + Relation ≈ Tail



- Push ***user vector*** + ***relation vector*** close to ***item vector***
- Measure closeness by Euclidean distance

TransRec (He et al, Recsys'17)

- Focused on next-item recommendation
  - Third-order relationship between <user, current item, next item>
  - Define relation vector as the current item:



User u_1
User u_2
User u_3

Translation operation:

$$\overrightarrow{prev.\ item} + \overrightarrow{user} \approx \overrightarrow{next\ item}$$

Predictive Model:

$$Prob(j \mid u, i) \propto \beta_j - d(\vec{\gamma}_i + \vec{T}_u, \vec{\gamma}_j),$$

Item bias

Translation distance

Figure 1: *TransRec* as a sequential model: Items (movies) are embedded into a 'transition space' where each user is modeled by a *translation* vector. The transition of a user from one item to another is captured by a user-specific translation operation.

Latent Relational Metric Learning (Tay et al, WWW'18)

- Distance-based predictive model:

$$s(p, q) = || p + r - q ||_2^2$$

where r is the latent relation vector, formed by an attentive sum over memory vectors:

$$r = \sum_i a_i m_i$$

Attentive weight, with inner product $s = p \odot q$ as input.
(the relation vector is dependent on user and item)

Memory vector, which can encode user attributes/interest.

LRML model:

$$s(p,q) = || p + r - q ||_2^2$$

Latent relation vector:

$$r = \sum_i a_i m_i$$

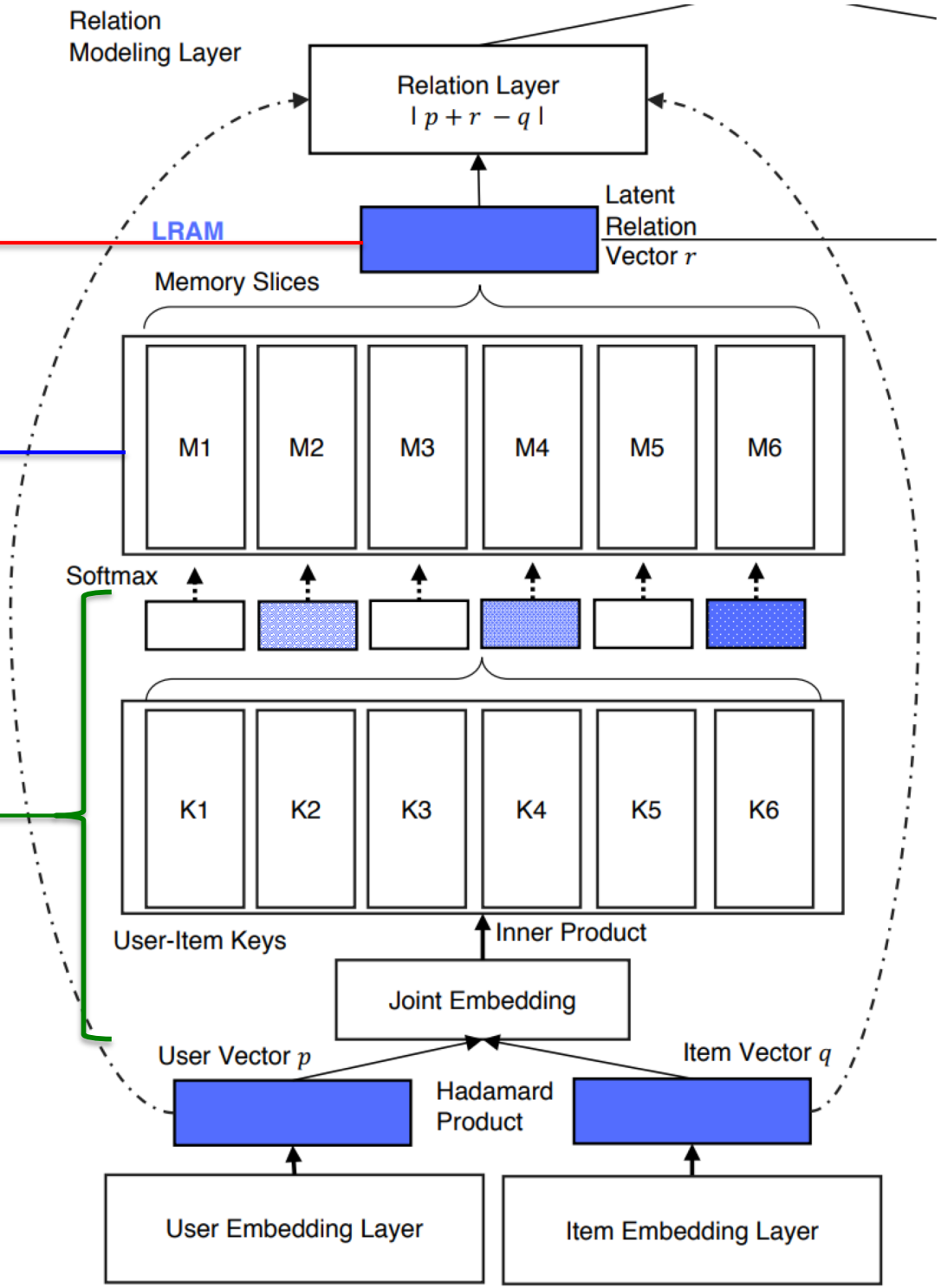
Memory vectors m_i (free parameters to learn)

Generate attentive weights:

$$a_i = (\mathbf{p} \odot \mathbf{q})^T \mathbf{k}_i)$$

Key for memory i

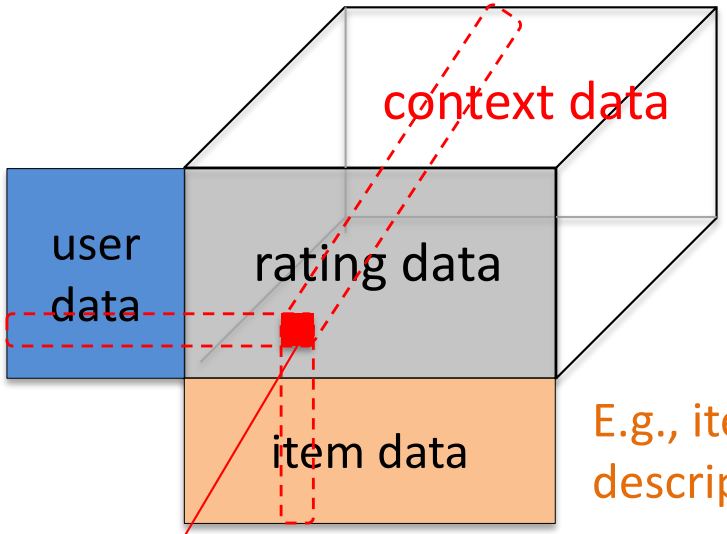
Normalize using softmax



Next: Methods of Matching Function Learning

1. CF models:
   - Based on Neural Collaborative Filtering (NCF) framework:
     NeuMF: Neural Matrix Factorization (He et al, WWW'17)
     ConvNCF: Outer Product-based NCF (He et al, IJCAI'18)
   - Based on Translation framework:
     TransRec: Translation-based Recommendation (He et al, Recsys'17)
     LRML: Latent Relational Metric Learning (Tay et al, WWW'18)
2. Feature-based models:
   - Based on Multi-Layer Perceptron:
     Wide&Deep (Cheng et al, DLRS'16),
     Deep Crossing (Shan et al, KDD'16)
   - Based on Factorization Machines (FM):
     Neural FM (He and Chua, SIGIR'17),
     Attentional FM (Xiao et al, IJCAI'17),
     DeepFM (Guo et al, IJCAI'17)

Recall: Input to Feature-based Models



E.g., user gender, age, occupation personality …

E.g., recent history, location, time, weather, mood …

E.g., item category, description, image …

Input Features:

1. Categorical features:
   user/item ID, bag-of-words, historical features…
2. Numerical features:
   textual/visual embeddings, converted features (e.g. TFIDF, GBDT)…

One-hot encoding

Each row encodes all info for a rating

Feature vector **x**

	A	B	C	…	TI	NH	SW	ST	…	TI	NH	SW	ST	…	Target y
	User				Movie					Other Movies rated					
$x^{(1)}$	1	0	0	…	1	0	0	0	…	0.3	0.3	0.3	0	…	5 $y^{(1)}$
$x^{(2)}$	1	0	0	…	0	1	0	0	…	0.3	0.3	0.3	0	…	3 $y^{(2)}$
$x^{(3)}$	1	0	0	…	0	0	1	0	…	0.3	0.3	0.3	0	…	1 $y^{(3)}$
$x^{(4)}$	0	1	0	…	0	0	1	0	…	0	0	0.5	0.5	…	4 $y^{(4)}$
$x^{(5)}$	0	1	0	…	0	0	0	1	…	0	0	0.5	0.5	…	5 $y^{(5)}$
$x^{(6)}$	0	0	1	…	1	0	0	0	…	0.5	0	0.5	0	…	1 $y^{(6)}$
$x^{(7)}$	0	0	1	…	0	0	1	0	…	0.5	0	0.5	0	…	5 $y^{(7)}$

Key to Feature-based Models

- Feature vector is high-dimensional but sparse
  - Consider the CF case: feature vector = user ID + item ID
  - Need to discover prediction patterns in nonzero features
- The interactions between features are important

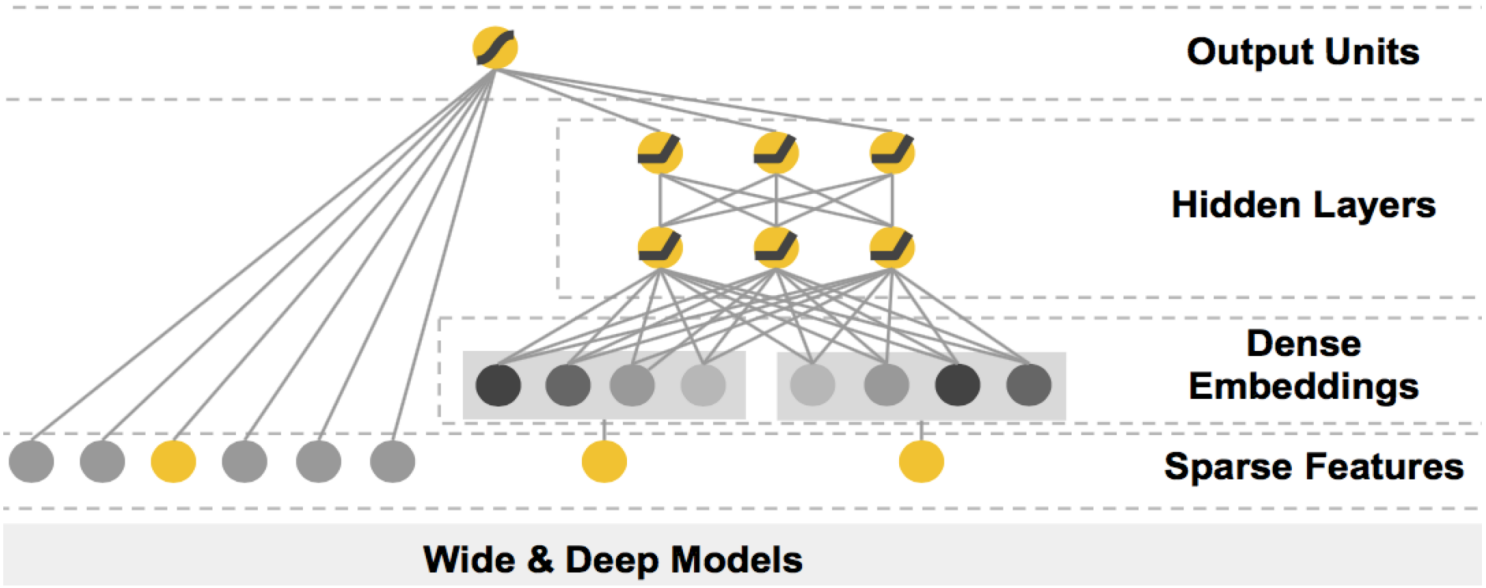
  E.g., users like to use food delivery apps at meal-time

  => Order-2 interactions between app category and time

  E.g., male teenagers like shooting games

  => Order-3 interactions between gender, age, and app category.
- Crucial for feature-based models to capture feature interactions (aka., cross features)

Wide&Deep (Cheng et al, Recsys'16)



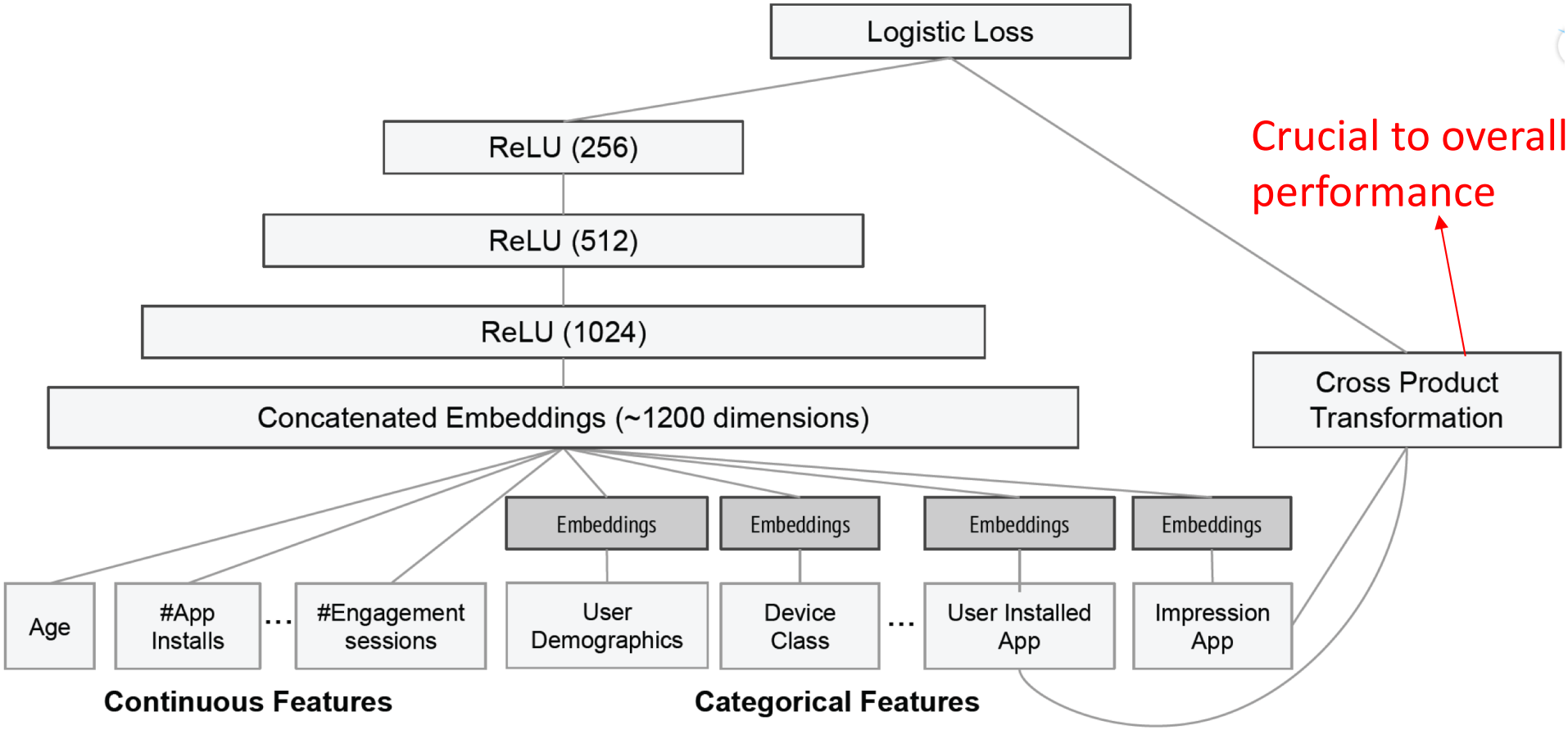
Standard MLP

Embedding Concatenation

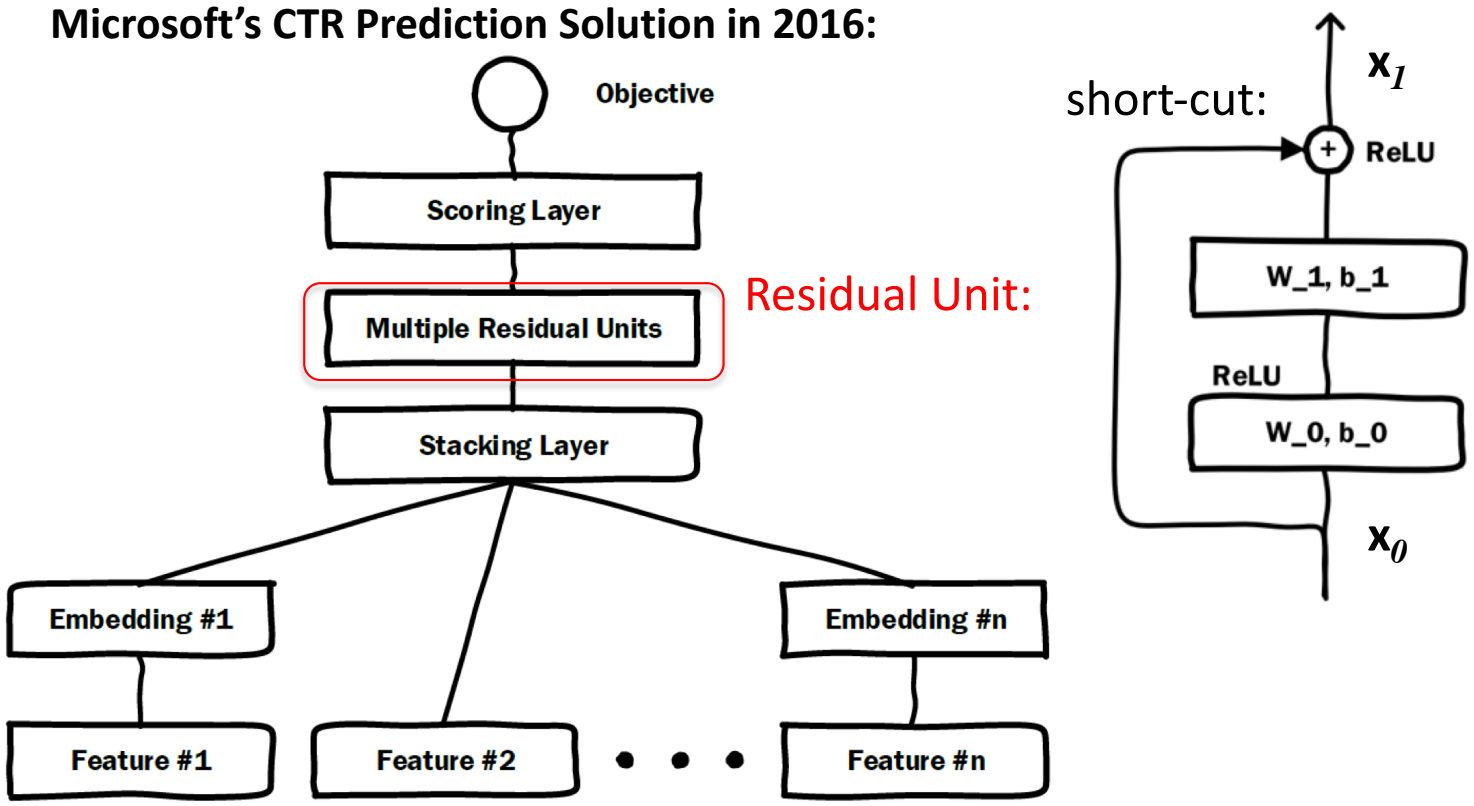
- The wide part is linear regression for memorizing seen feature interactions, which requires careful engineering on cross features.
  E.g., $AND(gender{=}female, language{=}en)$ is 1 iff both single features are 1

- The deep part is DNN for generalizing to unseen feature interactions.
  Cross feature effects are captured in an implicit way.

Wide&Deep for Google App Recommendation (Cheng et al, Recsys'16)



Deep Crossing (Shan et al, KDD'16)

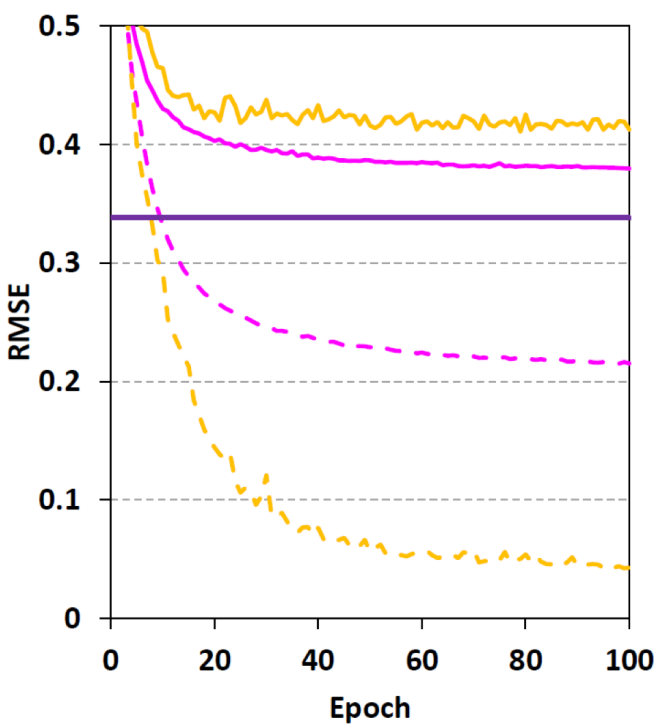


The main difference from Wide&Deep is the use of residual layers, which allow deeper network to be built (~10 layers).

Empirical Evidence (He and Chua, SIGIR'17)

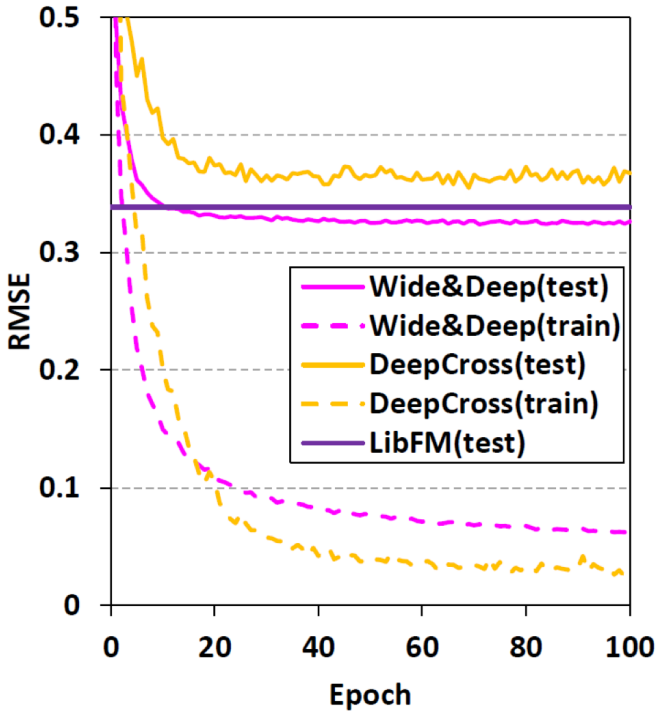
- However, when only raw features are used, both DL models don't perform well in learning feature interactions.

Solid line: testing loss; Dashed line: training loss



(a) Random initialization

With random initialization, deep methods underperform FM.



(b) FM as pre-training

With FM embeddings as pre-training, Wide&Deep slightly outperforms FM.

Some issues of DL methods:

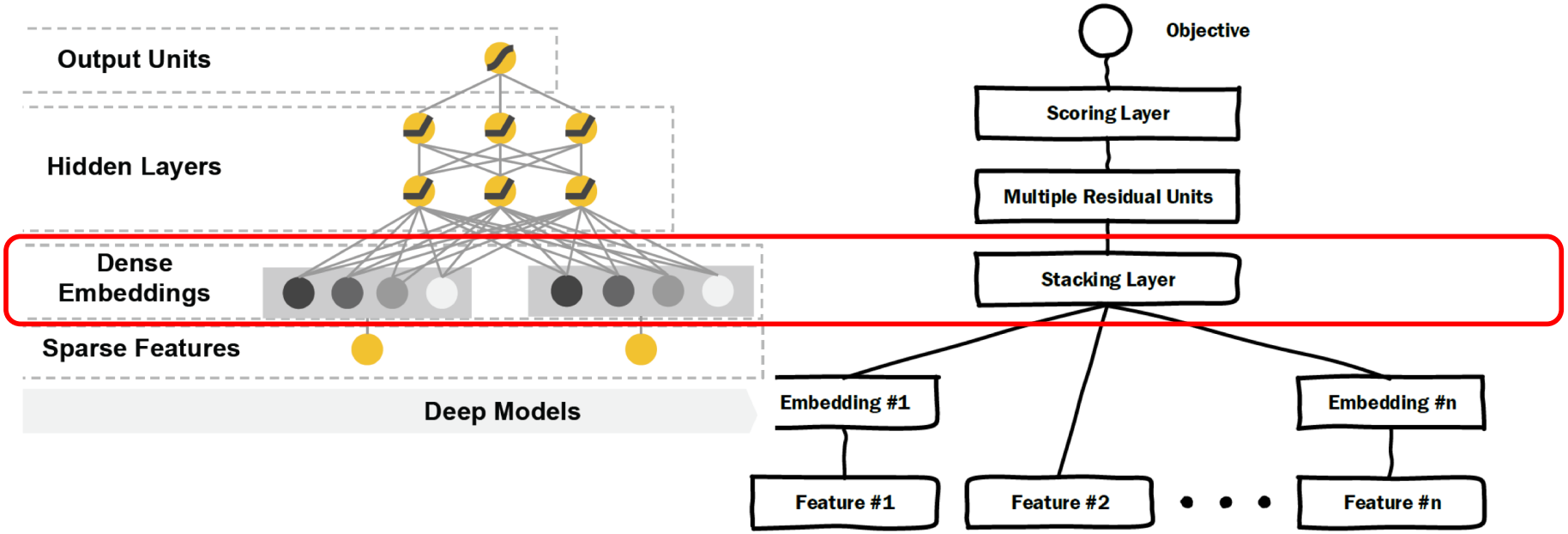
Easy to overfit

Hard to train well

Need good init.

Why MLP is Ineffective?

Besides optimization difficulties, one reason is in model design:



1. Embedding concatenation carries little information about feature interactions in the low level!
2. The structure of Concat+MLP is ineffective in learning the multiplicative relation (Beutel et al, WSDM'18).

Recap: Factorization Machine

- FM explicitly models second-order interactions between feature embeddings with inner product:

Only nonzero features are considered

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} < \mathbf{v}_i, \mathbf{v}_j > x_i x_j$$

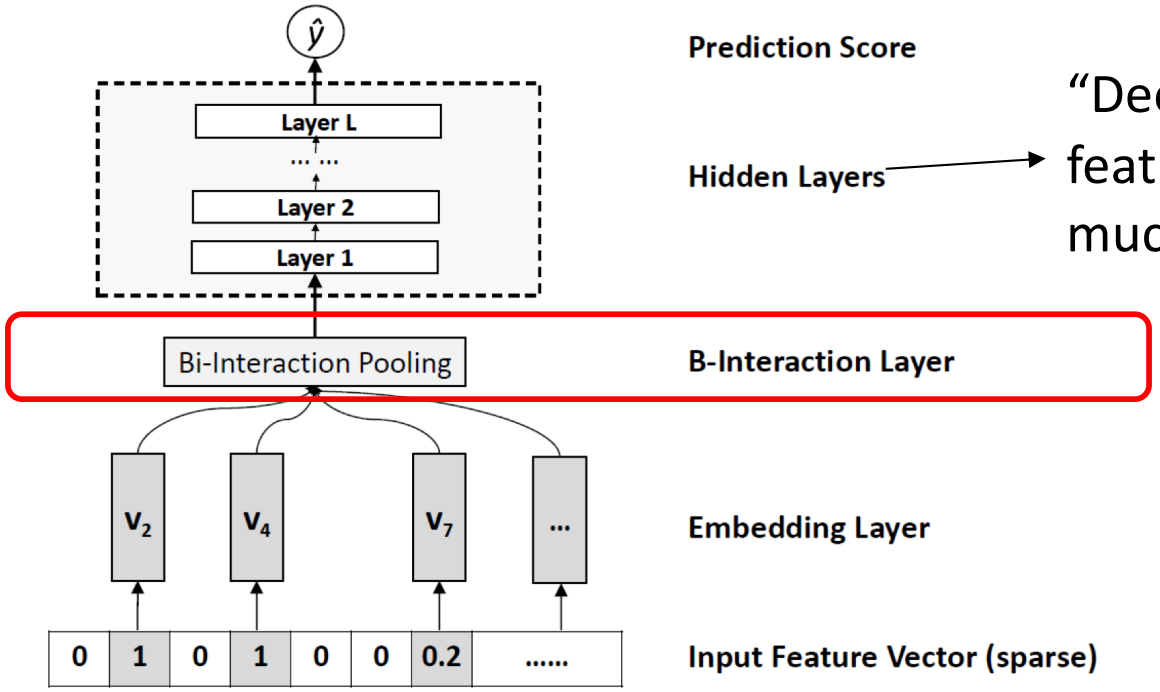
First-order: Linear Regression

Second-order: pair-wise interactions between nonzero features

  - Note: self-interaction is not included: $< \mathbf{v}_i, \mathbf{v}_i >$.

NFM: Neural Factorization Machine (He and Chua, SIGIR'17)

- Neural FM "deepens" FM by placing hidden layers above second-order interaction modeling.



"Deep layers" learn **higher-order** feature interactions only, being much easier to train.

Bilinear Interaction Pooling:

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

Experiment Results (He and Chua, SIGIR'17)

All methods are fed into raw features without any feature engineering

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recommendation
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**

1. Shallow embedding methods learn interactions, better than simple linear models

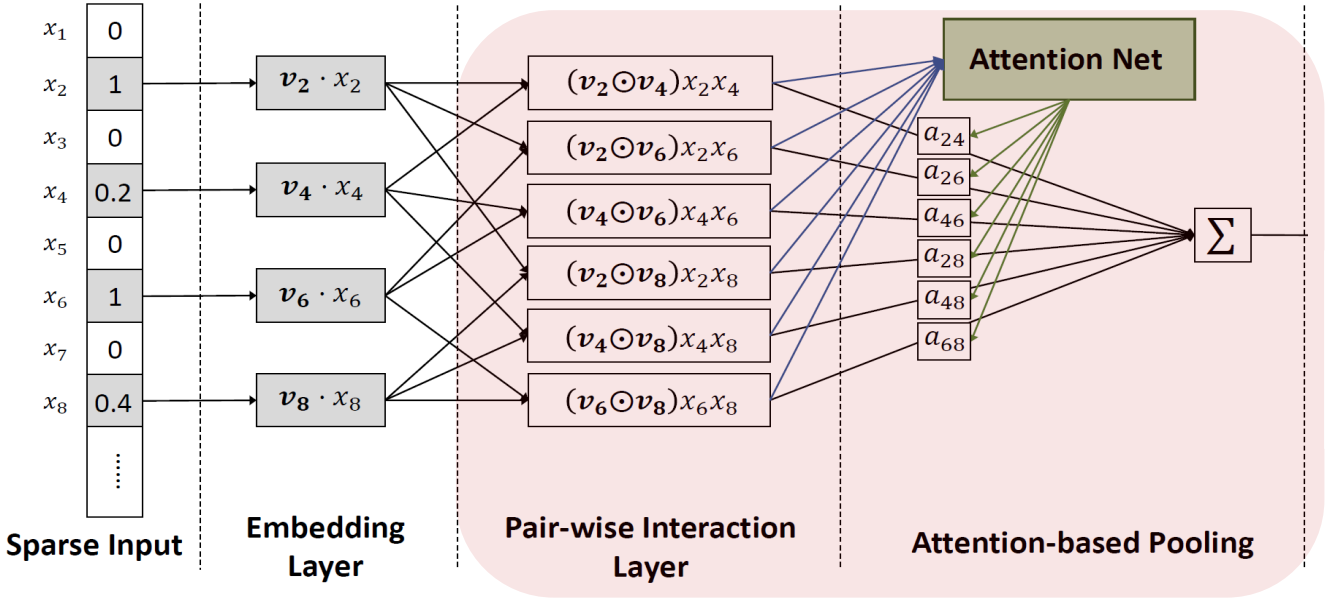
2. Deep embedding methods:
Wide&Deep = Concat+3 layers
DeepCross = Concat+10 layers

3. Neural FM
= BI pooling + 1 layer
Shallower but outperforming existing deeper methods with less parameters.

Codes: github.com/hexiangnan/neural_factorization_machine

AFM: Attentional Factorization Machine (Xiao et al, IJCAI'17)

- Neural FM treats all second-order feature interactions as contributing equally.
- Attentional FM uses an attention network to learn the weight of a feature interaction.



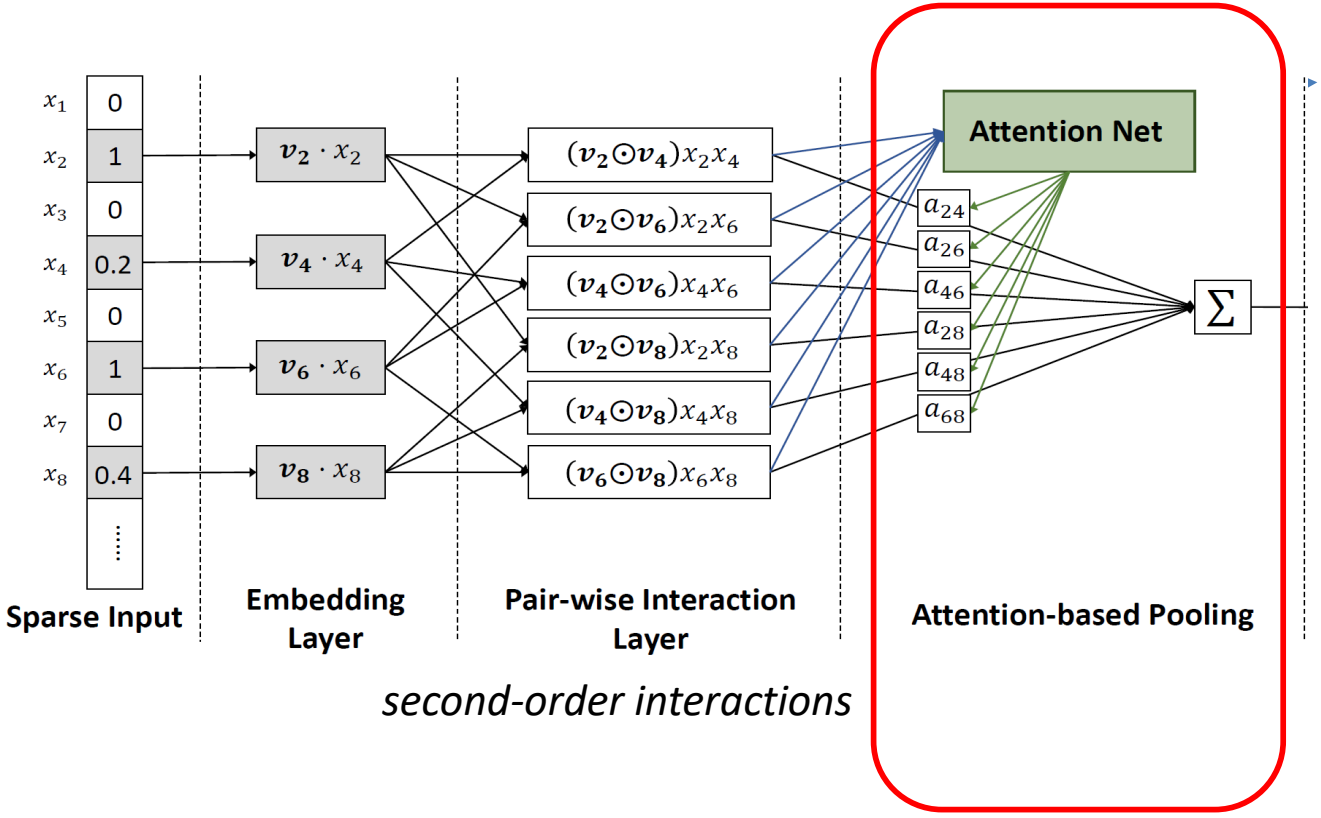
$$f_{ABI}(\mathcal{V}_x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(x_i\mathbf{v}_i \odot x_j\mathbf{v}_j)a_{ij}$$

$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j + \mathbf{b}),$$

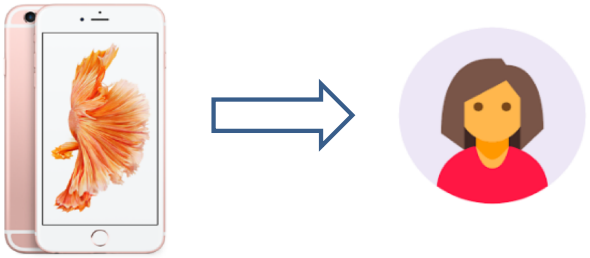
$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x}\exp(a'_{ij})},$$

Explaining Recommendation with AFM

The attention scores can be used to select the most predictive second-order feature interactions as explanations.



Example: explainable recommendation with second-order cross features:



<Female, Age 20>
<Age 20, iPhone>
<Female, Color Pink>
……

Experiment Results

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recommendation
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**
Attentional FM (0 layer)	**1.45M**	**0.3102**	**23.26M**	**0.4325**

AFM without hidden layers can be better than NFM with 1 hidden layer.

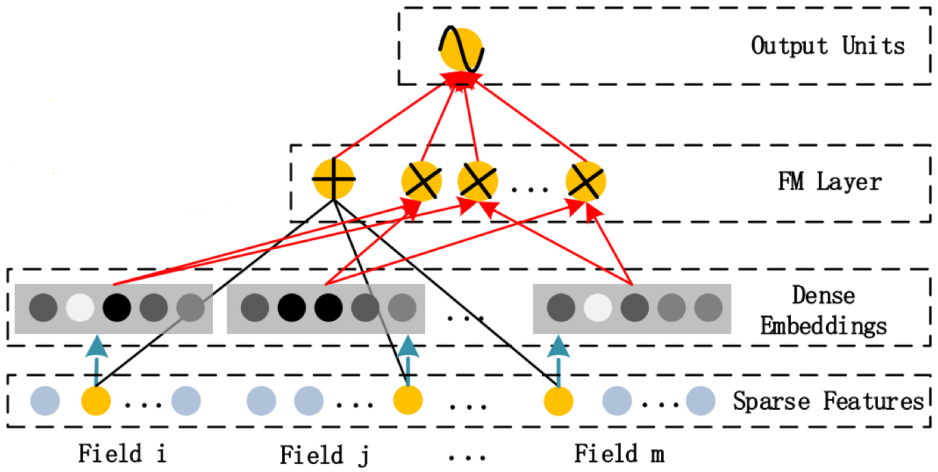
Codes: github.com/hexiangnan/attentional_factorization_machine

Adding hidden layers to AFM further improves.

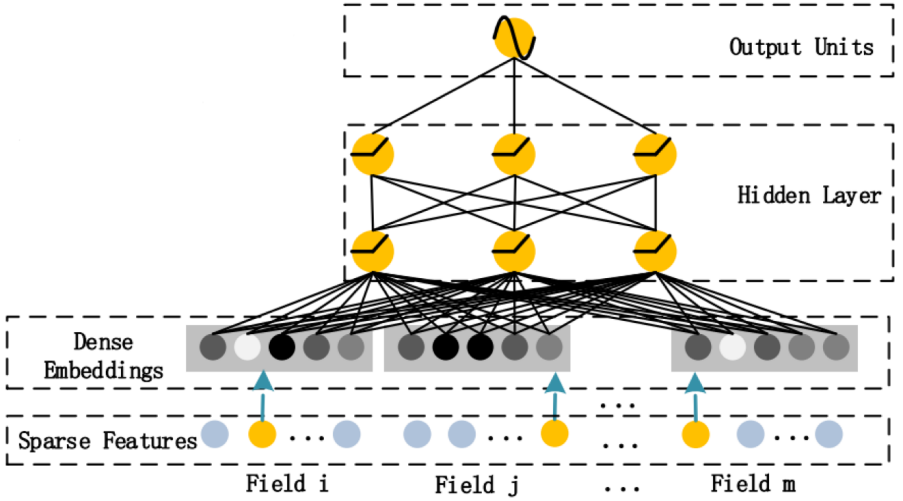
DeepFM (Guo et al., IJCAI'17)

- DeepFM ensembles FM and DNN and to learn both second-order and higher-order feature interactions:

FM structure:



DNN structure:



Prediction Model: $\hat{y}_{DeepFM} = \hat{y}_{FM} + \hat{y}_{DNN}$

- Note: FM and DNN share the embedding layer.
- DeepFM learns DNN from the residual of FM
- NeuralFM learns DNN based on the latent space of FM

Short Summary

- ✓ Feature interaction learning is crucial for matching function learning in recommendation.
  - Many models have been explored, e.g., DNN, FM, Attention Net etc.
- ✓ One insight is that doing early cross on raw features (or feature embeddings) is important to performance. E.g.,
  - Wide&Deep do manual cross on raw features
  - FM-based methods do second-order cross on feature embeddings
- Most models learn higher-order interactions with DNN, making higher-order effects hard to explain.
- It remains challenging to do higher-order interaction learning in an explainable way.

References

- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In Recsys 2016.
- Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In SIGIR 2017.
- Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. IJCAI 2017.
- Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In WWW 2015.
- Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In WSDM 2016.
- Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In CIKM 2015.
- Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In ICCV 2015.
- Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In SIGIR 2017.
- Fuzheng, Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In KDD 2016.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In WWW 2017.
- Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. CIKM 2017.

References

- Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Out Product-based Neural Collaborative Filtering. In IJCAI 2018.
- Tay, Yi, Shuai Zhang, Luu Anh Tuan, and Siu Cheung Hui. "Self-Attentive Neural Collaborative Filtering." *arXiv preprint arXiv:1806.06446* (2018).
- Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based Recommendation. In Recsys 2017.
- Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In WWW 2018.
- Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson et al. Wide & deep learning for recommender systems. In DLRS 2016.
- Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In KDD 2016.
- Xiangnan He, and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In SIGIR 2017.
- Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. IJCAI 2017.
- Guo, Huifeng, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. IJCAI 2017

Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
- Summary

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Summary

- Search and Recommendation are two sides of the same coin

  Search -> *Information Pull* with *explicit* info request (query)

  Recommendation -> *Information Push* with *implicit* info request (user profile, contexts)
- Technically, they can be unified under the same matching view
  - Though they are studied by different communities: SIGIR vs. RecSys
- Deep learning-based matching methods
  - Representation learning-focused
  - Matching function learning-focused
- Matching is a generic problem for a wide range of applications

  E.g., online advertising, question answering, image annotation, drug design

Challenges

- Data: building better benchmarks
  - Large-scale text matching data
  - Large-scale user-item matching data with rich attributes/contexts.
- Model: data-driven + knowledge-driven
  - Most current methods are purely data-driven
  - Prior information (e.g., domain knowledge, large-scale knowledge based) is helpful and should be integrated into data-driven learning in a principled way.
- Task: multiple criteria
  - Existing work have primarily focused on similarity
  - Different application scenarios should have different matching goals
  - Other criteria such as novelty, diversity, and explainability should be taken into consideration

Thanks!

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf